VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

Ostrava, 2010 **Bc. Marian Kuruc**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Systémy pohonu a energie pro mobilní robotické zařízení

Drive and Energy Systems for Mobile Robotic Device

Ostrava, 2010 **Bc. Marian Kuruc**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Zadání diplomové práce

Student: **Bc. Marian Kuruc**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Systémy pohonu a energie pro mobilní robotické zařízení
Drive and Energy Systems for Mobile Robotic Device

Zásady pro vypracování:

1. Výběr a realizace pohonu mobilního zařízení
2. Řízení pohonu zařízení.
3. Návrh správy napájení a dobíjení zařízení.
4. Návrh a realizace modulu řízení pohonu.
5. Platforma Freescale HCS12.
6. Využití vhodného OS (FreeRTOS, OSEK, ...)
7. Komunikační sběrnice CAN s aplikační vrstvou CANOpen.
8. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

1. BARRY, Richard. Using the FreeRTOS real time kernel : Practical guide., 2009. 164 s.
2. BARRY, Richard. FreeRTOS reference manual : API functions and configuration options., 2009. 119 s.
3. CAN in Automation e. V.. CANopen application layer and communication profile : CiA draft standard 301. 4th edition., 2002. 135 s.
4. Motorola, Inc. MC9S12DP256B : Device user guide. 2nd edition., 2005. 126 s.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Jiří Kotzian, Ph.D.**

Datum zadání: 20.11.2009

Datum odevzdání: 07.05.2010

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. Ing. Ivo Vondrák, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Marian Kuruc

Datum odevzdání diplomové práce: 7. 5. 2010

Poděkování

Děkuji vedoucímu diplomové práce Ing. Jiřímu Kotzianovi, Ph.D. za odborné konzultace a rady, které mi pomohly při vypracování této práce. Dále panu Ing. Vilému Srovnalovi za pomoc z oblasti teorie regulace. Dále Ing. Tomáši Solarskému za věcné rady při konstrukci měniče a v neposlední řadě pánům Hynku Prokopovi, Tomáši Lippovi a Jaromíru Konečnému, podílejících se na celkové realizaci mobilního robotického zařízení.

Abstrakt

Diplomová práce se zabývá návrhem řešení pohonu a správy energie mobilního robotického zařízení. To slouží jako průzkumné vozidlo pro průzkum životu nebezpečných oblastí. Má za úkol dojet do předem určeného cílového místa a vyhýbat se překážkám bez jakéhokoliv zásahu člověkem během jízdy. V diplomové práci je rozebrána problematika týkající se výběru pohonu a diskrétního regulátoru, ale také výběru akumulátoru a distribuce energie pro palubní elektroniku. S ohledem na použití koncepce distribuovaného řízení je část práce zaměřena na popis použitých komunikací a jejich komunikačních protokolů. Dále pak využití vhodného operačního systému reálného času. V práci je rozebrán jak celkový hardwarový návrh řešení, tak i softwarové řešení s využitím operačního systému reálného času s implementací komunikací a řídicích algoritmů.

Klíčová slova

FreeRTOS, CAN, CANopen, diskrétní PID regulátor, stejnosměrný motor, H-bridge, kvadraturní enkodér, digitální filtrace, mikrokontrolér, Freescale, HCS12, CodeWarrior

Abstract

This thesis deals with design solutions to drive and power management of mobile robotic device. Mobile robotic device serves as a reconnaissance vehicle for the exploration of life-threatening areas. Its task is to reach a predetermined destination and avoid obstacles without human intervention during the journey. It also analyzed issues related to the choice of drive and discrete controller, but also the selection and distribution of battery power for onboard electronics. In view of the concept of using distributed control is part of the work focused on the description of communications and communication protocols. Then use the appropriate real time operating system. In this thesis we discuss how the overall hardware solutions as well as software solutions, using real-time operating system with the implementation of communication and control algorithms.

Keywords

FreeRTOS, CAN, CANopen, discrete PID controller, direct current motor, H-bridge, quadrature encoder, digital filtering, microcontroller, Freescale, HCS12, CodeWarrior

Seznam použitých symbolů a zkratek

ADC	–	(Analog-Digital Converter), analogově digitální převodník
BDM	–	(Background Debugger Module), rozhraní pro programování mikroprocesorů
BLDC	–	(BrushLess DC), synchronní elektromotor, stejnosměrný elektromotor s elektronickým komutátorem
C	–	Programovací jazyk
CAN	–	(Controller Area Network), lze volně přeložit jako datová sběrnice místní sítě
CANopen	–	Nástavba sběrnice CAN, definuje aplikační vrstvu
CC – CV	–	(Constant Current – Constant Voltage), způsob nabíjení akumulátoru metodou konstantní napětí – konstantní proud
CPU	–	(Central Processing Unit), procesor
CRC	–	(Cyclic Redundancy Check), Cyklický redundantní součet
DPS	–	Deska Plošných Spojů
ECT	–	(Enhanced Capture Timer), rozšířený modul čítače / záchytného systému
EEPROM	–	(Electronically Erasable Programmable Read-Only Memory), elektricky mazatelná paměť
EMI	–	(ElectroMagnetic Interference), elektromagnetické rušení
FIFO	–	(First In, First Out), speciální volatilní paměť
FIR	–	(Finite Impulse Response), filtr s konečnou impulzní odezvou
GSM	–	(Groupe Spécial Mobile), globální Systém pro Mobilní komunikaci
GPS	–	(Global Positioning Systém), systém globální navigace
HCS12	–	Označení výrobní řady mikroprocesorů firmy Freescale
I2C	–	(Inter Integrated Circuit), datová sběrnice
IGBT	–	(Insulated Gate Bipolar Transistor), bipolární tranzistor s izolovaným hradlem
kbps	–	(KiloBits Per Second), jednotka rychlosti datového přenosu
LCD	–	(Liquid Crystal Display), displej z tekutých krystalů
Mbps	–	(Millions of Bits Per Second), jednotka rychlosti datového přenosu
MCU	–	(MicroController Unit), mikrokontrolér, mikropočítač
MOSFET	–	(Metal Oxide Semiconductor Field Effect Transistor), polem řízený tranzistor
MSCAN	–	(Motorola Scalable Controller Area Network), komunikační rozhraní zabudované v mikroprocesorech Freescale
NASA	–	(National Aeronautics and Space Administration), národní úřad pro letectví a kosmonautiku
NMT	–	(Network Management) , zpráva definovaná standardem CANopen
OS	–	Operační systém
PC	–	(Personal Computer), osobní počítač
PDO	–	(Proces Data Objects), zpráva definovaná standardem CANopen
PID	–	Proporcionální – Integrační – Derivační regulátor
PSD	–	Proporcionální – Sumační – Diferenční regulátor
PWM	–	(Pulse Width Modulation), pulzně šířková modulace
RAM	–	(Random Access Memory), paměť s libovolným (náhodným) přístupem
RTOS	–	(Real-Time Operating System), operační systém reálného času
RC	–	(Remote Control), označení pro rádiem řízené modely
SYNC	–	(Synchronization) , zpráva definovaná standardem CANopen
SMD	–	(Surface Mount Device), součástka určená pro povrchovou montáž

SMT	–	(Surface Mount Technology), technologie povrchové montáže součástek
Through – hole	–	Technologie montáže elektronických součástek
TTL	–	(Transistor Transistor Logic), technologie používaná k výrobě integrovaných obvodů
UART	–	(Universal Asynchronous Receiver Transmitter), univerzální asynchronní sériové rozhraní
USB	–	(Universal Serial Bus), univerzální sériová sběrnice pro připojení periferií k počítači
Wi-Fi	–	(Wireless Fidelity), komunikační standard pro lokální bezdrátové sítě

Obsah

1 Úvod

Žijeme v době nevídaného technického pokroku. Každý měsíc se představují veřejnosti nové technické vymoženosti a technologie. Tuto dobu někdo nazývá informační, jiný digitální. Tato dvě slova ale znamenají jedno: svět, ve kterém žijeme, se mění. Není to změna, která by trvala několik desítek let, jak by se mohlo zdát. Je to změna, která čím dál rychleji mění náš pohled na techniku.

Jedním z úkolů, o který se člověk snaží již delší dobu je sestrojit vozidlo, které by se dokázalo pohybovat zcela autonomně a vyhýbalo se případným objektům v jeho okolí. Doplněním navíc o orientaci v terénu a bezdrátovou komunikaci bychom dostali zcela autonomní mobilní jednotku. Ta by nacházela uplatnění v průzkumných, záchranných a vyprošťovacích akcích a obecně v prostředí nebezpečném pro člověka. Využití můžeme nalézt ale i v každodenním životě.

Myšlenka není nikterak nerealizovatelná, neboť již v dnešní době existují například autobusy, které se dokážou po cestě pohybovat bez zásahu řidiče. Příkladem může být město Eindhoven, které má vybudovanou síť speciálních vozovek s vestavěnými čidly, které dovolují pohyb autobusů bez zásahu řidiče. Vice na [27] . Dalším příkladem může být soutěž DARPA urban challenge, při které zcela autonomní vozidla projíždějí zadanou trasu ulicemi města. [28]

Vznikl tedy nápad na sestrojení laboratorního modelu průzkumného vozidla, které by aspoň částečně bylo autonomní. Dokázalo by se na základě údajů z čidel a senzorů vyhýbat překážkám před a za automobilem a umělo dojet na požadované místo. Dalším z požadavků bylo, aby se rozdělení palubní elektroniky podobalo rozdělení elektroniky v autě. To znamená, že celá elektronika průzkumného vozidla by tvořila distribuovaný řídicí systém, kde by jednotlivé celky byly propojeny proti rušení odolnou sběrnicí. Tímto nápadem vznikly celkově čtyři diplomové práce řešící vždy určitou část průzkumného vozidla.

Diplomová práce navazuje na předchozí zkušenosti při konstrukci podobného průzkumného vozidla v rámci bakalářské práce, které je popsáno v kapitole 2. Při tvorbě nového vozidla bude potřeba zjistit nedostatky stávajícího řešení a navrhnout řešení nové za účelem jejich odstranění. Toto je shrnuto v kapitole 3.

Kapitola 4 je věnována návrhu řešení. Je zde navrženo celkové blokové schéma průzkumného vozidla, včetně popisu funkcí jednotlivých modulů a vyznačení datových komunikací. Dále jsou zde uvedeny požadavky a návrh řešení modulů popisovaných v této diplomové práci.

Pro pojezd vozidla a řízení jeho rychlosti je potřeba správně navrhnout pohon a způsob jeho regulace. Kapitola 5 provází návrhem pohonu od jeho samotného výběru dle typu činnosti motoru, výpočtu na základě parametrů podvozku, hardwarovou realizaci měniče, až po samotný návrh softwarového regulátoru.

Důležitou částí každého mobilního zařízení je akumulátor elektrické energie. Kapitola 6 pojednává o jeho výběru, možnostech nabíjení a hlídání stavu akumulátoru a v neposlední řadě o filtracích potřebných pro měření analogových veličin akumulátoru.

Kapitola 7 se zaobírá hardwarovým návrhem modulů. Jsou zde uvedeny návrhy schémat zapojení a plošných spojů modulu pohonu a modulu správy energie.

Kapitola 8 pojednává o výběru operačního systému reálného času, jeho výhodách a nevýhodách. Součástí kapitoly je také popis synchronizačních nástrojů a úprava operačního systému pro daný mikrokontrolér.

K distribuovanému řízení patří bezesporu otázka výběru vhodných datových komunikací. Ta je spolu s popisem komunikačních protokolů a implementací v rámci operačního systému uvedena v kapitole 9.

Kapitola 10 rozebírá návrh softwarového řešení modulů. Jsou zde uvedeny algoritmy a popisy jednotlivých vláken.

Kapitola 10 shrnuje dosažené výsledky této diplomové práce.

V pořadí poslední je kapitola 11. Ta obsahuje celkové zhodnocení práce a návrhy na vylepšení.

2 Současný stav

Před započetím práce bylo k dispozici vozidlo, které je zobrazeno na obrázku Obr. 1 a které bylo vytvářeno v rámci bakalářských prací čtyř studentů. Toto vozidlo sloužilo k prvnímu seznámení s problematikou a poskytlo cenné informace a zkušenosti, které budou využity pro konstrukci nového modelu robotického zařízení. Vozidlo je vyrobeno z malého RC modelu s elektrickým pohonem a je doplněno o vývojové moduly s procesory HCS12. Tyto moduly tvoří distribuovaný řídicí systém, který vozidlo řídí. Zároveň je vozidlo osazeno senzory, které poskytují potřebné informace o prostředí.

Obr. 1 Současný stav robotického zařízení

Tento první prototyp mobilního zařízení je funkční, ale při jeho konstrukci se vyskytly problémy, které budou při vypracování diplomových prací řešeny a odstraněny.

- Pohon vozidla je navržen pro vysoké rychlosti a nelze jej kvalitně regulovat na malé rychlosti.
- Díky použitému jednokanálovému inkrementálnímu snímači není znám směr otáčení motoru.
- Jsou použity celé vývojové moduly, které nejsou hardwarově uzpůsobené pro vozidlo.
- Vozidlo je příliš malé a neumožňuje montáž všech potřebných modulů a senzorů.
- Vozidlo postrádá rychlé a kvalitní senzory pro monitorování terénu.
- Vozidlo nemá (kromě relativně nepřesné GPS) informaci o absolutní pozici.
- Bezdrátová komunikace je řešena pomocí GSM sítě, která vyžaduje SIM kartu a složité připojování pomocí mobilního operátora.

3 Cíle

Diplomová práce navazuje na předešlý vývoj autonomního průzkumného vozidla v rámci bakalářských prací. Takovéto průzkumné vozidlo najde uplatnění všude, kde je potřeba provést rekognoskaci prostředí a zásah bez intervence člověka. Příkladem tak jsou místa s výskytem nebezpečných látek, ale také autonomní řízení dopravních prostředků (kyberauta). [29]

Každé autonomní vozidlo řeší základní otázky, „Kde jsem?“, „Kam se chci dostat?“ a „Jak se tam dostanu?“. Aby vozidlo bylo schopno si odpovědět na tyto otázky, musí být schopno vnímat svoje prostředí, lokalizovat se v něm a naplánovat cestu. K tomu mu poslouží kvalitní senzory pro měření vzdáleností překážek od vozidla s možností jízdy i za snížené nebo nulové viditelnosti.

Celkovým cílem práce je sestrojit model autonomního vozidla, které bude těchto úkonů schopno alespoň do jisté míry. Vytvořit tak platformu pro výzkum a vývoj s využitím moderních technologií a implementovat je do problematiky autonomního řízení. Vozidlo bude koncipováno jako distribuovaný systém řízení. Hlavním rysem takového systému je jeho modulárnost, na kterou bude při návrhu brán zřetel. Navržený model bude výsledkem týmové práce čtyř diplomantů.

Bude nutné vytvořit celkovou strukturu průzkumného vozidla a jeho rozčlenění na jednotlivé řídicí moduly dle jejich funkce. Rovněž bude potřeba zvolit vhodný typ datové komunikace mezi zařízeními a komunikační protokol (standard). Pro jednotlivé řídicí moduly bude potřeba zvolit vhodné mikrokontroléry. S ohledem na rozsáhlost návrhu a multiplatformnost, bude vybrán vhodný operační systém reálného času a následně implementován pro danou architekturu mikrokontroléru.

Cílem této práce je navrhnout a sestrojit moduly pro správu energie a pohonu zařízení. Tyto budou základními moduly na vozidle, ovládající pohon průzkumného vozidla a distribuci elektrické energie pro palubní elektroniku. Přihlíženo zde bude na využití moderních technologií a řešení.

Úkolem modulu správy energie bude distribuce energie pro celou palubní elektroniku. Palubní elektronika bude rozdělena do tří samostatných napájecích okruhů. Jejich odepínáním, tedy zmenšením odběru proudu, bude možno ušetřit energii akumulátoru. Dalším úkolem modulu bude monitoring velikosti napětí akumulátoru a odběru proudu a hlídání stavu akumulátoru během provozu. Bude mít rovněž na starost řízení dobíjení akumulátoru ze zdroje nabíjení. Zdrojem nabíjení může být buď klasický laboratorní zdroj s patřičnou velikostí napětí a proudu, nebo také alternativní zdroj energie, jako například solární panel. V rámci práce na modulu bude potřeba vyřešit:

- návrh schéma zapojení, výběr vhodných komponent, návrh desky plošných spojů
- výběr vhodného akumulátoru
- nabíjení a sledování stavu akumulátoru
- distribuce elektrické energie pro palubní elektroniku
- návrh a implementace řídicích algoritmů
- nasazení operačního systému reálného času
- výběr komunikace modulu s okolím a vytvoření komunikačních protokolů

Úkolem modulu pohonu bude starost o pohon průzkumného vozidla. To v sobě zahrnuje možnost plynulé regulace otáček a dále možnost změny směru otáčení pohonu za pomocí diskrétního regulátoru. Pro potřeby zpětné vazby bude nutné vyřešit snímání otáček a směru otáčení motoru. Při správném hardwarovém návrhu měniče bude možné při brzdění elektromotorem využít i rekuperaci a tím dobíjet zdroj elektrické energie. Bude zde také vyřešena i mechanická brzda pro případ dlouhodobého stání vozidla. Z důvodů velkých proudů tekoucích motorem bude nutné zvážit potřebu potenciálového oddělení signálové a silové země. Potřebné signály pro buzení výkonového stupně by pak byly odděleny optočlenem. V rámci práce na modulu bude potřeba vyřešit:

- návrh schéma zapojení, výběr vhodných komponent, návrh desky plošných spojů
- výběr a dimenzování pohonu vozidla
- zpracování dat z kvadraturního enkodéru
- návrh a implementace řídicích algoritmů
- nasazení operačního systému reálného času
- výběr komunikace modulu s okolím a vytvoření komunikačních protokolů

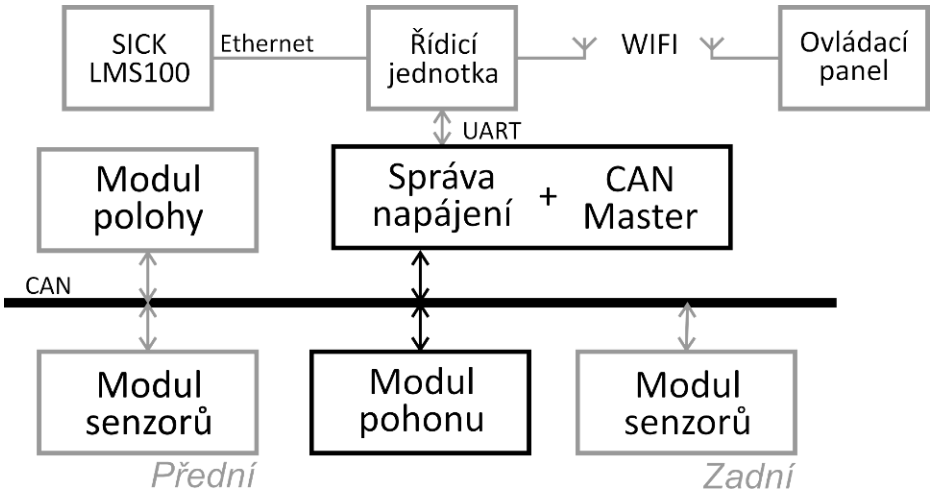
Výběr operačního systému reálného času, návrh komunikací a tvorba komunikačních protokolů je společnou prací dvou diplomantů. Samotná implementace v rámci jednotlivých modulů je pak již předmětem samostatné práce v této diplomové práci.

4 Návrh řešení

V kapitole bude popsán celkový návrh koncepce průzkumného vozidla. Na tuto navazuje další podkapitola, zabývající se podrobněji částí řešení v rámci této diplomové práce.

4.1 Blokové schéma průzkumného vozidla

Řízení průzkumného vozidla je koncipováno jako distribuovaný řídicí systém. Ten je složen z osmi modulů, sedm z nich je umístěno přímo na vozidle a komunikují spolu prostřednictvím sběrnice CAN. Osmou část tvoří vzdálený systém s vizualizací, který komunikuje s vozidlem prostřednictvím sítě Wi-Fi a slouží pro sledování vozidla a jeho manuální ovládání. Na obrázku Obr. 2 vidíme blokový návrh řešení rozčlenění palubní elektroniky na jednotlivé moduly dle jejich funkce.



Obr. 2 Blokové schéma průzkumného vozidla

Popis jednotlivých modulů:

- **Řídicí jednotka** - Jejím úkolem je navázání spojení se vzdáleným ovládacím panelem, výběr a výpočet nejvhodnější trasy k zadanému cíli, vyhodnocení překážek před a za vozidlem. Prostřednictvím sběrnice ethernet je k jednotce připojen laserový scanner SICK LMS100 pro měření vzdálenosti překážek kolem vozidla. Řízení vozidla se bude provádět prostřednictvím zadávání příkazů ostatním (podřízeným) modulům.

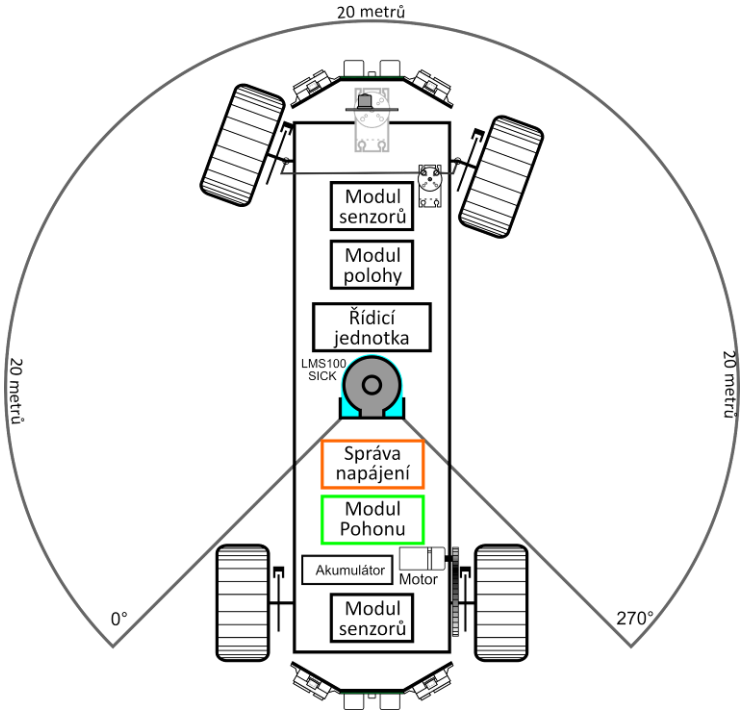
- **Správa napájení –** Slouží pro distribuci elektrické energie z akumulátoru do jednotlivých modulů na vozidle. Řídí nabíjení, dobíjení a sleduje stav akumulátoru při vybíjení a dále jej vyhodnocuje. Je také datovým mostem mezi CAN sběrnicí a nadřazeným systémem.

- **Modul polohy** - Poskytuje řídící jednotce informace o aktuální pozici vozidla. Tu vypočítává z globálních a relativních informací o poloze z připojených senzorů. Mimo to získává informace o okolním prostředí jako teplota, tlak, koncentrace výbušných látek**.**

- **Modul senzorů** – Na vozidle se nacházejí dva tyto moduly. Ty na vozidle fungují jako sběrné uzly, které shromažďují informace ze svého okolí a ty poskytují ostatním modulům. Modul nacházející se na přední části vozidla zároveň ovládá natočení předních kol.

- **Modul pohonu** – Obsahuje výkonovou část pro řízení pohonu. Implementuje algoritmus pro regulaci otáček motoru, proudové omezení, ovládání brzdy. Požadovaná rychlost jízdy je ovládána prostřednictvím příkazů z nadřazeného systému.

- **Ovládací panel** - Vizualizace a správa vozidla. Data z vozidla jsou přenášena prostřednictvím sítě Wi-Fi a použita pro kontrolu, zadávání nových parametrů a jejich vizualizaci na dotykovém LCD displeji.



Obr. 3 Rozmístění komponent na vozidle

Na obrázku Obr. 3 vidíme navržené rozmístění komponent na průzkumném vozidle. Moduly jsou umístěny tak, aby se omezila délka signálových vedení k jejich senzorům a aktuátorům.

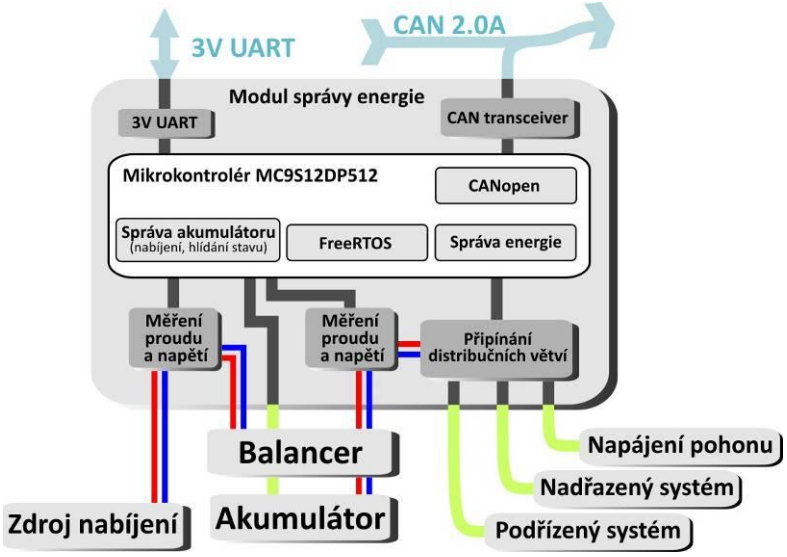
Moduly správa napájení a modul pohonu, které jsou na obrázku Obr. 2 zvýrazněny, budou dále popsány v následující kapitole.

4.2 Blokové schéma modulu správy energie a modulu pohonu

V této kapitole bude stručně popsán návrh řešení obou modulů. Podrobnější řešení je pak uvedeno v následujících kapitolách diplomové práce.

4.2.1 Modul správy energie

Modul správy energie je základním modulem na průzkumném vozidle, starající se o distribuci elektrické energie pro ostatní palubní elektroniku. Ta je rozdělena do tří skupin. První skupina je napájení pohonu, kde jsou velké proudové odběry a v případě potřeby je proto možné napájení pohonu odepnout. Další skupinu tvoří nadřazený systém, který je spolu s další skupinou, podřízeným systémem, odepínán pouze během nabíjení akumulátoru. Samotné připínání distribučních větví je realizováno elektronickými spínači, tedy tranzistory. Navržená struktura modulu je vidět na obrázku Obr. 4.



Obr. 4 Blokové schéma modulu správy energie

Stav akumulátoru je během jízdy vozidla hlídán pro ochranu proti nadproudu a podpětí. K tomu slouží blok měření proudu a napětí. Při zjištění vybití akumulátoru, se připojí zdroj nabíjení, jehož nabíjecí napětí a nabíjecí proud je rovněž monitorován.

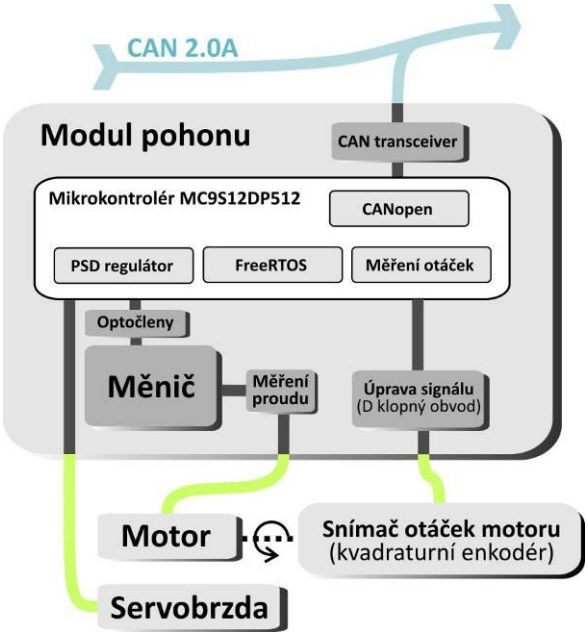
Vhledem k použitému akumulátoru Li-pol je potřeba vyřešit obvod pro vyrovnání napětí na jednotlivých článcích akumulátorové baterie. K tomu slouží obvod zvaný balancer. Algoritmus nabíjení je implementován v mikrokontroléru a řídí regulaci nabíjecího proudu, sleduje stav napětí na jednotlivých článcích akumulátorové baterie a detekuje ukončení nabíjení. Obvod balanceru a algoritmus pro nabíjení akumulátoru je prací jiného studenta v rámci jeho bakalářské práce.

Modul správy energie tvoří datový most mezi podřízeným a nadřazeným systémem. Komunikace s podřízeným systémem je umožněna prostřednictvím sběrnice CAN, na které je implementován komunikační protokol CANopen. Komunikace s nadřazeným systémem je umožněna pomocí 3V rozhraní UART, na kterém je vytvořen vlastní komunikační protokol. Data

z podřízeného systému jsou tak prostřednictvím modulu správy energie dostupná pro nadřazený systém a naopak.

4.2.2 Modul pohonu

Úkolem modulu pohonu je řízení pojezdu průzkumného vozidla. Na obrázku Obr. 5 vidíme jeho zjednodušené blokové schéma.



Obr. 5 Blokové schéma modulu pohonu

Pro pojezd průzkumného vozidla je vybrán SS motor s permanentními magnety, jehož buzení obstarává výkonový měnič, označovaný jako H-bridge. Proud motoru je pro potřeby ochrany motoru proti nadproudu měřen snímačem na Hallově principu. Výkonová část modulu pohonu je od signálové části galvanicky oddělena pomocí optočlenů. Regulace pohonu je pak zajištěna diskrétním PID regulátorem, označovaný též jako PSD regulátor. Akční veličina PSD regulátoru pak následně ovlivňuje PWM výstup mikrokontroléru a tím řídí měnič.

Pro dekódování údajů kvadraturního enkodéru je využito klopného obvodu typu D. Jeho vhodným zapojením je možné ze dvou signálů fázově posunutých o 90° získat údaj o směru otáčení motoru. Tento signál spolu s obdélníkovým signálem z enkodéru vstupuje do záchytného systému mikrokontroléru, kde se dále zjišťují aktuální otáčky motoru.

Modul také disponuje PWM výstupem pro ovládání servobrzdy. Ta je aktivována v případě požadavku velmi rychlého zastavení vozidla a v případě delší doby stání vozidla.

Pro komunikaci s ostatními moduly je použita sběrnice CAN s aplikační vrstvou CANopen.

Při návrhu obou modulů je využito operačního systému reálného času FreeRTOS. Ten zajišťuje multiplatformnost řešení v případě použití jiné architektury mikrokontroléru. Zároveň poskytuje mnoho výhod a ulehčuje tak návrh řídicího software.

5 Řízení pohonu zařízení

Podvozek průzkumného vozidla je založen na RC modelu Baja 5B SS, který je standardně vybaven spalovacím motorem o výkonu 26 ccm. Ten je však pro pohon průzkumného vozidla nevhodný. Vzhledem k určenému prostředí (převážně interiér) a snazšímu a výhodnějšímu použití byl zvolen elektromotor. [13]

5.1 Výběr pohonu

Důležitým faktorem pro výběr motoru je jeho způsob ovládání, krouticí moment, výkon a otáčky za minutu. S ohledem na bateriový provoz průzkumného vozidla zde můžeme zařadit také účinnost motoru.

Existuje několik typů motorů v závislosti na jejich činnosti a mechanické konstrukci. Mezi nejčastěji používané patří tyto:

a) Asynchronní střídavý motor

Je to nejrozšířenější pohon v elektrotechnice vůbec. Tok energie mezi točivým polem statoru a většinou klecovým vinutím rotoru je realizován výhradně pomocí elektromagnetické indukce, proto se často tento motor označuje jako motor indukční. Točivé pole rotoru je oproti točivému poli statoru opožděno, vzniká takzvaný skluz. Rychlost se opět řídí změnou frekvence. Pro zajištění konstantního krouticího momentu při změně frekvence je potřeba měnit i efektivní hodnotu napětí. [22]

- Vysoká spolehlivost, jednoduchá konstrukce
- Pro výkony od 100 W do řádově MW, napájení z běžné střídavé sítě
- Nevýhody: složitější řízení a hardware měniče

b) Krokový bipolární a unipolární motor (Stepper Motor)

Krokové motory se podle konstrukce rozdělují na dvě varianty, bipolární a unipolární. U bipolárního motoru je stator podobný jako u BLDC motoru, ale běžně obsahuje pouze 2 vinutí (dva páry napájecích vodičů). Rotor je tvořen permanentními magnety s póly (zuby), které jsou přitahovány "zuby" statorového vinutí. Přesnost krokování je dáno počtem zubů rotoru (někdy jich obsahuje i několik stovek). Motor dokáže zastavit a zůstat stát na konkrétní pozici zubu nebo na 1/2 pozici (mezi zuby), přičemž síla držení motoru na místě je dána velikostí proudu. Pro jedno celé otočení (jednu otáčku) je nutné výrazně více sekvencí než u BLDC motoru = složitější řízení a zpracování. [22]

- Jednoduché řízení a hardware budiče, Přesné polohování rotoru
- Nevýhody: nevýhodný poměr výkonu (krouticího momentu) vůči hmotnosti motoru; s rostoucí rychlostí rotoru klesá krouticí moment

c) Bezkartáčový stejnosměrný motor (BLDC Motor)

Bezkartáčový BLDC motor (Brushless DC motor) patří do kategorie stejnosměrných, i když je strukturou podobný střídavému 3fázovému synchronnímu motoru. Z tohoto důvodu nelze

na motor připojit přímo stejnosměrné napětí ze zdroje, ale je nutné provádět jeho spínání. Stator je běžně tvořen 3 budícími vinutími zapojené do hvězdy. Rychlost otáčení se řídí frekvencí (časováním) sekvencování, zatímco střídou a frekvencí impulsů uvnitř sekvencí se určuje "hladkost" běhu. Z tohoto pohledu je řízení v otevřené smyčce (bez zpětné vazby) problematické, zvláště když se motor má rozbíhat již zatížen. Proto se obvykle využívá uzavřené regulační smyčky, kde se pozice (natočení) určuje buď měřením proudu Hallovým senzorem nebo absolutním či inkrementálním rotačním enkodérem. Točivý moment je definován velikostí proudu, přičemž vysokonapěťové impulsy vytvářejí stejný efekt. [22]

- Velká účinnost, spolehlivost a životnost
- Absence komutátoru a tím i jiskření produkující rušení
- Nevýhody: složitější řízení a hardware měniče

d) Kartáčový (komutátorový) stejnosměrný motor (Brushed DC Motor)

Princip tohoto motoru je založen na periodickém střídavém přepínání polarity napájecího stejnosměrného proudu pomocí komutátoru po každém otočení rotoru o 180°. Rotor je tvořen elektromagnety navinutými okolo pólových nástavců. Jejich nejběžnější počet je 3, aby se minimalizovala možnost zaseknutí. Rychlost otáčení se řídí velikostí napětí, nejjednodušeji jeho spínáním (PWM modulace napájecího napětí). Toto řízení generuje šum, který je nutné filtrovat nebo se využívá vysoké frekvence spínání. Kartáčové komutátorové motory mají omezenou účinnost díky komutátoru, který limituje max. proud a napětí. [22]

- Jednodušší návrh hardware měniče pro pohon
- Momentová charakteristika přímo úměrná proudu motoru
- Využití momentu již od malých otáček
- Velký rozsah otáček motoru
- Změna otáček motoru velikostí napětí
- Změna směru otáčení pomocí změny polarity
- Nevýhoda – přítomnost mechanického komutátoru, vznik elektromagnetického rušení v důsledku jiskření na komutátoru

Ze všech možných typů činnosti motorů byl vybrán kartáčový stejnosměrný motor s permanentními magnety.

Mezi špičkové stejnosměrné pohony patří pohony švýcarského výrobce Maxon, od kterého byl vybrán pohon i pro naši aplikaci. Výrobce dodává ke svým motorům také převodovky a enkodéry. O kvalitě značky Maxon vypovídá spolupráce s NASA a použití těchto motorů pro vesmírné sondy na Marsu. Jejich nevýhodou je velká pořizovací cena.

Výhody pohonu Maxon:

- Velmi kvalitní mechanické zpracování, Vysoká účinnost motoru až 88 %
- Kovové kartáče s delší životností oproti grafitovým, rotor je tvořen samonosnou cívkou
- Rotor neobsahuje feromagnetické jádro, to vede k podstatně menšímu vzniku oblouků a tím i elektromagnetického rušení při přepínání na komutátoru, delší životnost komutátoru
- Možnost až osminásobného přetížení, limitováno teplotou rotorového vinutí [23] [11]

5.1.1 Postup výpočtu při výběru motoru

S výběrem a návrhem celého pohonu ochotně pomohl český distributor Maxonu, firma Uzimex. Nejprve bylo nutné specifikovat požadavky pro pohon. Zde jsou uvedeny základní vlastnosti:

- Hmotnost vozidla $m_V = 20\ kg$, pohon zadní nápravy
- Stálý převod kol 16:1, poloměr kola $r = 0{,}1\ m$
- Požadovaná rychlost vozidla $v \leq 3\ m/s$
- napájení motoru 12 V

Pro výpočet bylo využito návodu ve formátu *.xls firmy Uzimex, uvedeném v elektronické příloze na CD pod číslem [9].

V prvním kroku si vypočteme potřebný moment setrvačnosti vozidla. Pro případ vozidla lze aplikovat vzorec pro výpočet momentu setrvačnosti válce (1), kde hmotnost $m = m_V / 2$ a poloměr kola $r = 0{,}1\ m$.

$$J = \frac{1}{2} m r^2 = \frac{1}{2} \cdot 10 \cdot 0{,}1^2 = \underline{\underline{0{,}05\ kgm^2}} \tag{1}$$

Pokud známe moment setrvačnosti, můžeme vypočítat potřebný moment motoru dle vzorce (3). K tomu je potřeba zvolit požadovanou dobu zrychlení $\Delta t = 1\ s$. Dále pak požadované otáčky Δn motoru, které dostaneme ze známé hodnoty požadované rychlosti vozidla a poloměru kola (2).

$$\Delta n = \frac{60 \cdot v}{2\pi r} = \frac{60 \cdot 3}{2\pi \cdot 0{,}1} = \underline{\underline{286\ ot/\min}} \tag{2}$$

Potřebný moment pohonu je dán vztahem (3).

$$M' = J \cdot \frac{\pi}{30} \cdot \frac{\Delta n}{\Delta t} = 0{,}05 \cdot \frac{\pi}{30} \cdot \frac{286}{1} = \underline{\underline{1{,}497\ Nm}} \tag{3}$$

Protože se na vozidle nachází stálá převodovka s převodovým poměrem $p_1 = 16:1$, minimální požadovaný krouticí moment motoru bude v ideálním případě 16x menší (4).

$$M'' = \frac{M'}{16} = \underline{\underline{93{,}59\ mNm}} \tag{4}$$

Stejnosměrný motor s takovým krouticím momentem M'' a při zadaných požadavcích firma Maxon ve svém výrobním programu nenabízí. Proto byl podle krouticího momentu M'' a otáček motoru Δn z katalogu Maxon vybrán motor RE 40 a k němu odpovídající převodovka GP 42 C s převodovým poměrem $p_2 = 4{,}3:1$. Celá sestava je vidět na obrázku Obr. 6. Tato sestava disponuje celkovým krouticím momentem dle vztahu (5).

$$M = M_M \cdot p_2 = 0{,}0896 \cdot 4.3 = \underline{\underline{385\ mNm}} \tag{5}$$

Ze vztahu (5) je patrné, že navržený pohon má velkou rezervu oproti požadovanému krouticímu momentu M'', a to více než čtyřnásobnou.

Obr. 6 Pohon s motorem Maxon RE40

a) Motor RE 40

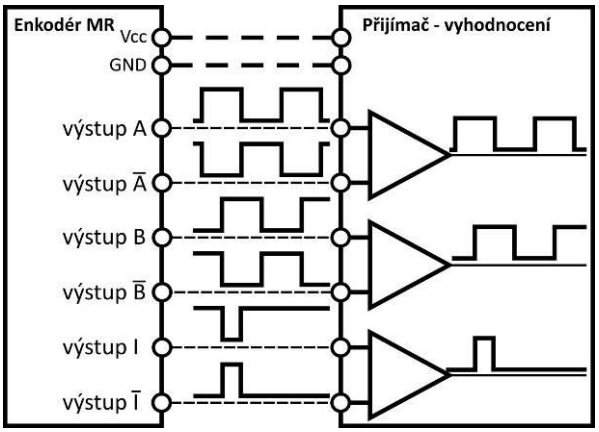
- Výkon 150 W při jmenovitém napětí 12 V
- Točivý moment 89,6 mNm, jmenovitý proud 6 A
- Otáčky naprázdno 6920 otáček/min, proud naprázdno 241 mA
- Maximální účinnost 88,4 % [24]

b) Převodovka GP 42 C

- Jednostupňová planetová převodovka s převodovým poměrem 4,3 : 1
- Výsledný točivý moment 385 mNm při napětí 11,7 V a proudu 5,44 A [25]

c) Enkodér MR typ L

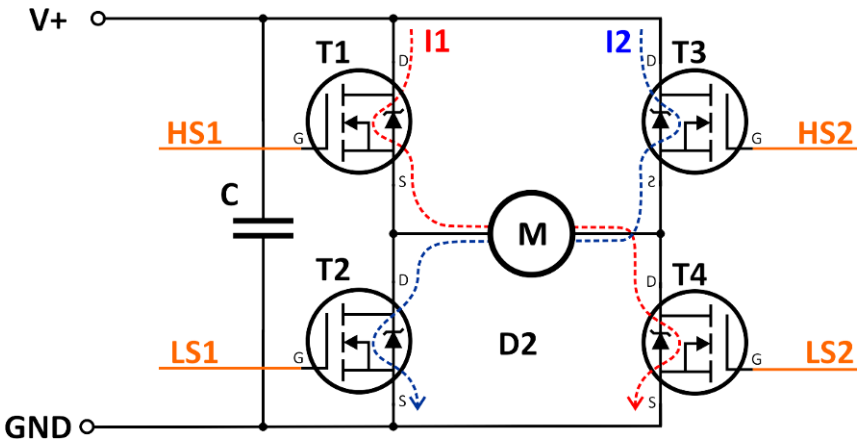
- Enkodér s kvadraturním výstupem s 256 impulsy na otáčku, značeny jako kanály A a B
- Obsahuje i indexový výstup - jeden impuls na otáčku pro zjištění polohy hřídele motoru, značí se písmenem I
- Všechny tři kanály jsou z enkodéru vedeny jako diferenční signály tak, jak je uvedeno na obrázku Obr. 7. [26]



Obr. 7 Diferenční výstup z inkrementálního snímače MR [26]

5.2 Výběr měniče pro pohon

Výběr typu měniče je podmíněn výběrem typu motoru. Pro vybraný stejnosměrný motor je možno použít buď jednokvadrantový nebo čtyřkvadrantový pulzní měnič. Byl vybrán čtyřkvadrantový pulzní měnič, nazývaný také jako H-bridge. Ten má oproti jednokvadrantovému měniči výhodu elektronické změny směru otáčení motoru.



Obr. 8 H-bridge

Na obrázku Obr. 8 vidíme zjednodušené schéma H-bridge. Motor se roztočí, sepnou-li se tranzistory T1 a T4 (signály HS1 a LS2). Motorem bude protékat proud I1. Pokud naopak otevřeme tranzistory T2 a T3 (signály HS2 a LS1) bude se motor otáčet v opačném směru než v předchozím případě. Motorem pak poteče proud I2.

Zapojení dále poskytuje možnost brzdění. To se provádí sepnutím tranzistorů T2 a T4 (signály LS1 a LS2), při němž se zkratuje rotorové vinutí motoru a tím dojde k rychlejšímu zastavení motoru.

Výhodou zapojení na obrázku Obr. 5 je také možnost rekuperace energie do meziobvodu. Ten je nejčastěji tvořen kondenzátorem nebo o mnoho lepším superkondenzátorem. Ten na rozdíl od akumulátorů dokáže uchovat energii v elektrické formě a na rozdíl od kondenzátoru disponuje mnohokrát větší měrnou energií (Wh/kg). Akumulovaná energie se pak využije pro pokrytí velkých proudových odběrů např. při rozběhu motoru.

Jako spínací prvek se používají N MOSFET tranzistory s nízkým odporem v sepnutém stavu $R_{DS(on)}$. Výběr P MOSFET tranzistorů s velmi malým odporem $R_{DS(on)}$ je podstatně horší než u N kanálových. Z toho důvodu se pak pro horní větve H-bridge (T1 a T3) používají rovněž N MOSFET tranzistory. Ty se však musí spínat napětím o velikosti $V_{MOTOR} + V_{GS}$. Z toho vyplývá použití budiče pro horní tranzistory, který zajistí jejich korektní spínání.

K měniči se obvykle dává i senzor pro měření proudu ať už z důvodu omezení maximálního proudu do motoru, nebo jako vstupní veličina regulátoru. S výhodou lze použít senzory pracující na principu Hallova jevu. Na těchto snímačích nevzniká napěťový úbytek.

5.3 Výběr regulátoru

Pro ovládání pohonné jednotky je potřeba vymyslet způsob jejího řízení. Nabízí se v podstatě dvě kritéria podle čeho pohon regulovat, a tím je regulace na konstantní otáčky nebo regulace na konstantní moment. Obojí má své výhody i nevýhody, které můžeme shrnout v následujícím přehledu:

a) Regulace na konstantní otáčky

Nejpoužívanější způsob regulace pohonu, neboť akční veličina je odvozena z aktuální hodnoty otáček pohonu a při vzniku regulační odchylky se regulátor snaží otáčky udržet. Výsledkem je snaha regulátoru udržet stálé požadované otáčky při proměnném zatížení pohonu. Tento typ regulátoru ale nedbá na značné proudové odběry, které mohou nastat při velkém zatížení pohonu a mohou vést až ke zničení akumulátoru.

b) Regulace na konstantní moment

Tento typ regulace se používá spíše jako doplňkový a to pro omezení maximálního proudového odběru. Mechanický točivý moment motoru se vypočte dle vztahu (6). [10]

$$M \cong M_{em}, \quad M = c \cdot \phi \cdot I_a \,, \tag{6}$$

kde

M - mechanický točivý moment

M_{em} - elektromagnetický moment stroje

c - konstrukční konstanta stroje

ϕ - magnetický tok ve vzduchové mezeře mezi statorovým a rotorovým vinutím

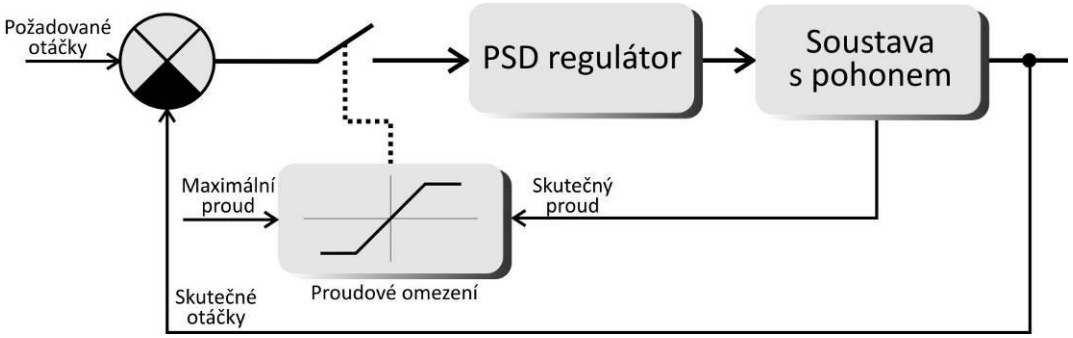
I_a - proud vinutím kotvy stroje

Mechanický točivý moment M na hřídeli stroje je v případě, že zanedbáme moment mechanických ztrát roven elektromagnetickému momentu M_{em}. Dále je ze vztahu patrné, že při konstantním magnetickém toku ϕ (tzn. u motoru s permanentními magnety) je mechanický točivý moment motoru přímo úměrný proudu vinutím kotvy I_a. Z toho vyplývá, že pro regulaci na konstantní moment je potřeba znát hodnotu proudu odebíraného pohonem.

Z předchozího odstavce je tedy zřejmé, že při regulaci na konstantní moment nemůže dojít k přetížení motoru, neboť právě regulátorem je limitována maximální hodnota odběru proudu. Toto nelze použít pro regulaci, protože průzkumné vozidlo pohybující se v určitém prostředí musí být schopno zdolávat překážky a to se děje právě na úkor zvětšení proudového odběru. Nicméně tento typ regulátoru použijeme jako pomocný v případě, že dosáhneme limitující hodnoty odběru proudu.

Nejpropracovanějším řešením je použití kaskádního zapojení obou výše jmenovaných regulátorů.

Pro regulaci pohonu vozidla byla použita regulace na konstantní otáčky s předřazenou limitací proudu, jak je znázorněno na následujícím obrázku Obr. 9.



Obr. 9 Schéma návrhu regulace pohonu

Pro samotnou implementaci regulátoru může být využito jak analogového tak diskrétního regulátoru. Protože je k dispozici výkonný 16bitový mikroprocesor, je použita diskrétní verze PID regulátoru označovaná jako PSD regulátor.

5.4 Diskrétní PID regulátor

Bezpochyby nejrozšířenějším typem regulátoru je v současnosti diskrétní varianta PID regulátoru (dále jen PSD). Tento Proporcionálně – Sumačně – Diferenční regulátor dokáže velmi kvalitně provádět regulační pochody a navíc od analogové verze jednoduše implementovat různé typy filtrací měřených veličin, které mohou být zašuměny.

Na rozdíl od spojitých regulátorů, které okamžitě reagují na změnu regulační odchylky, se u diskrétních toto děje v časových intervalech označovaných jako vzorkovací perioda T_v.

Jednotlivé složky PSD regulátoru jsou:

a) Proporcionální

Proporcionální složka je tvořena prostým zesilovačem se zesílením K_p. Výstup z P regulátoru je přímo úměrný regulační odchylce do něj vstupující. Samotný P regulátor nedokáže odstranit poruchu v podobě trvalé regulační odchylky.

b) Sumační

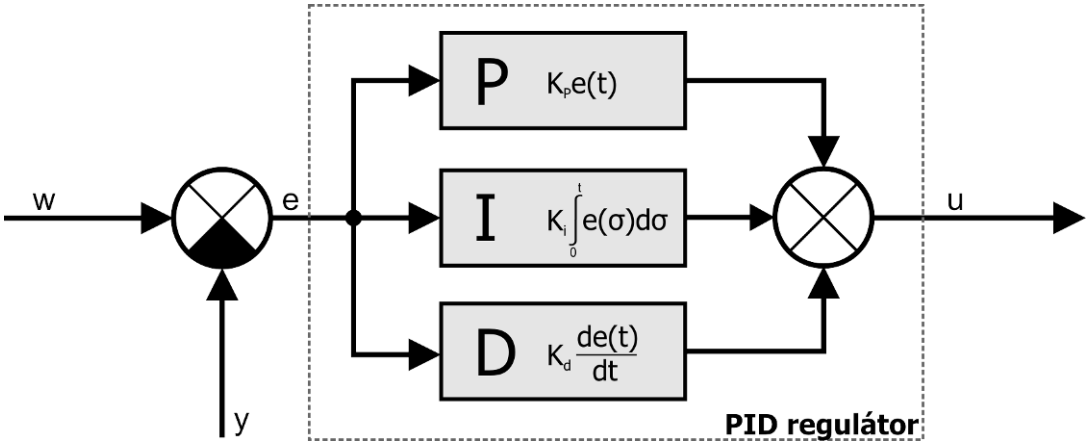
Sumační složka je tvořena sumou všech regulačních odchylek v předchozích krocích ($k \cdot T_v$) a současnou hodnotou regulační odchylky. Suma je poté vynásobena integrační časovou konstantou T_i. Výhodou S regulátoru je to, že dokáže úplně eliminovat regulační odchylku, nevýhodou je pak zpomalení regulačního děje a zhoršení stability.

Dalším nevýhodným jevem je tzv. windup, který nastává v případě dlouhotrvající regulační odchylky, kdy výsledek sumace roste nade všechny meze a poté způsobuje nežádoucí prodloužení regulačního pochodu. Podrobné řešení bude rozebráno v některé z následujících kapitol.

c) Diferenční

Diferenční složka je tvořena diferencí aktuální a předchozí hodnoty regulační odchylky. Diference je poté vynásobena diferenční časovou konstantou T_d. Diferenční regulátor zrychluje regulační děj a také zesiluje šum. Proto se často před regulátor zařazuje dolnopropustný filtr. Samotný regulátor tohoto typu se nikdy nepoužívá. [7]

Pro regulaci pohonu byl použit PSD regulátor. Ačkoliv by pro stejnosměrný motor postačil PS regulátor, pro názornost byl přesto použit celkový PSD algoritmus.



Obr. 10 Paralelní PID regulátor [7]

Na obrázku Obr. 10 vidíme PID regulátor v paralelním tvaru. Výstupem regulátoru je akční veličina *u*, která je dána vztahem:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\sigma)d\sigma + T_d \frac{de(t)}{dt} \right) \qquad (7)[7]$$

Pro diskretizaci vztahu (7) a vytvoření vztahu pro PSD regulátor je potřeba zvolit vhodnou diskrétní aproximaci integrační a derivační složky. Pro diskrétní integraci (sumaci) může být použita aproximace buď obdélníkovou, nebo přesnější lichoběžníkovou metodou. Přepis vztahu pro lichoběžníkovou metodu pak bude následující (8).

$$\int_0^t e(\sigma)d\sigma \approx T_v \frac{1}{2}\sum_{k=0}^{n}\left(e(k) + e(k-1)\right) \qquad (8)$$

Pro diskrétní derivaci je přepis jednodušší a spočívá v rozdílu hodnot regulační odchylky minulé od nynější (9).

$$\frac{de(t)}{dt} \approx \frac{e(n) - e(n-1)}{T_v} \qquad (9)$$

Derivační složka je velice citlivá na kolísání (šum) regulační odchylky. Proto se před derivační článek obvykle řadí dolnopropustný filtr n-tého řádu, který při vhodně zvolené frekvenci zlomu potlačí nežádoucí vysoké frekvence. [7]

Návrh filtrace bude popsán v kapitole *6.3 Filtrace analogových veličin* modulu správy energie, kde je také využíván.

Výstupem PSD regulátoru s použitím výše uvedených vzorců bude hodnota akční veličiny v n-tém kroku dle vztahu (10). Vztahu (10) se také říká polohový tvar PSD regulátoru. [8]

$$u(n) = K_p\left(e(n) + \frac{T_v}{T_i}\sum_{k=0}^{n}\left(e(k)\right) + \frac{T_d}{T_v}\left(e(n) - e(n-1)\right)\right) \tag{10}$$

kde

K_p - proporcionální zesílení

T_i - integrační (sumační) časová konstanta [s]

T_d - derivační (diferenční) časová konstanta [s]

T_v - vzorkovací perioda [s]

5.5 Návrh PSD regulátoru

Pro použitý stejnosměrný motor s permanentními magnety a inkrementálním kvadraturním enkodérem ve spojení se čtyřkvadrantovým pulzním měničem bude navrhnut PSD regulátor. Postup se dá shrnout do následujících kroků [6] :

a) Odměření přechodové charakteristiky motoru při zátěži – Na motor je přiveden v čase $t = 0$ jednotkový skok v podobě dané hodnoty napájecího napětí a z dat inkrementálního senzoru dostaneme závislost otáček v čase t.

b) Identifikace soustavy v Matlabu – Z přechodové charakteristiky je určen v Matlabu pomocí nástroje *ident* operátorový přenos $G_S(s)$.

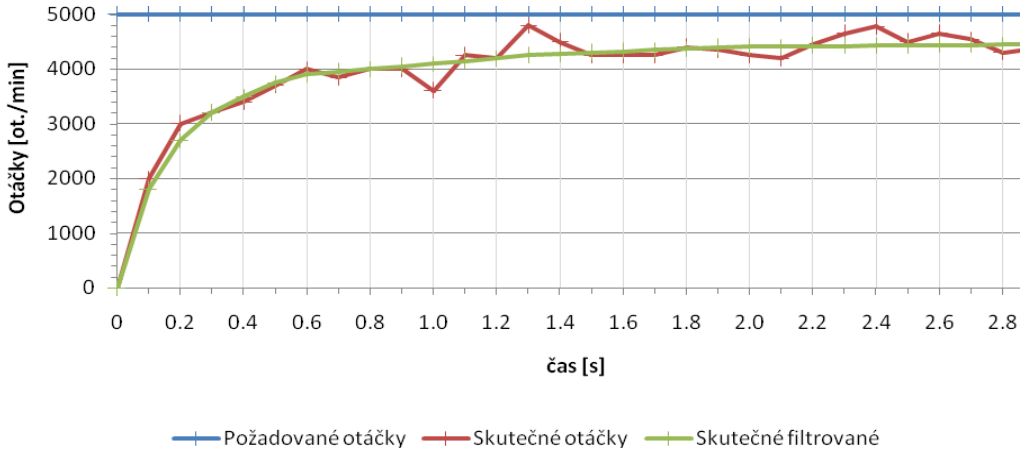
c) Diskretizace přenosu $G_S(s)$ **–** Při diskretizaci přenosů je zadávána zvolená perioda vzorkování T_v.

d) Vytvoření PSD algoritmu v jazyce C – Vytvoření rutiny pro PSD regulátor, která se bude provádět cyklicky s periodou T_v.

e) Návrh regulačního obvodu v Simulinku a ladění konstant PSD regulátoru – Na soustavu necháme působit PSD regulátor vytvořený v jazyce C a importovaný jako S funkce do Simulinku. Postupně budeme zadávat konstanty Kp, Ti a Td a zjišťovat, jaký mají vliv na kvalitu a rychlost regulace a na míru citlivosti na šum. Experimentálním postupem jsou tedy zjištěny výsledné konstanty K_p, T_i, T_d pro zadanou vzorkovací periodu T_v.

5.5.1 Změření přechodové charakteristiky motoru

Na 12V motor maxon byl přiveden jednotkový skok v podobě napětí o velikosti $U_{STEP} = 8{,}57\,V$ odpovídající rychlosti 5000 otáček/min. Záchytným systémem mikrokontroléru HCS12 jsou měřeny časové intervaly mezi jednotlivými impulzy z enkodéru, následně přepočítány na ot/min. Pomocí USB BDM programátoru / debuggeru jsou data posílána do PC, kde jsou zobrazeny v aplikaci Freemaster. Výsledný graf je vidět na obrázku Obr. 11.



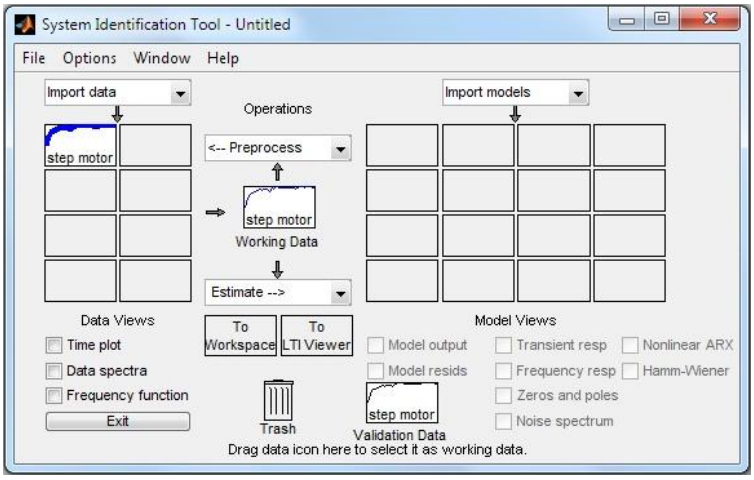
Obr. 11 Graf – Změřená odezva motoru na jednotkový skok

Z grafu lze vidět, že naměřené otáčky kolísají vlivem odporových sil (třecí síly, valivý odpor atd.) v daném okamžiku měření. Toto bude potřeba zohlednit při navrhování konstant regulátoru tak, aby nedocházelo k nadměrné citlivosti regulátoru na toto kolísání. Dále také regulátor odstraní regulační odchylku v ustáleném stavu, kdy by se aktuální otáčky měly rovnat požadovaným otáčkám 5000 otáček/min.

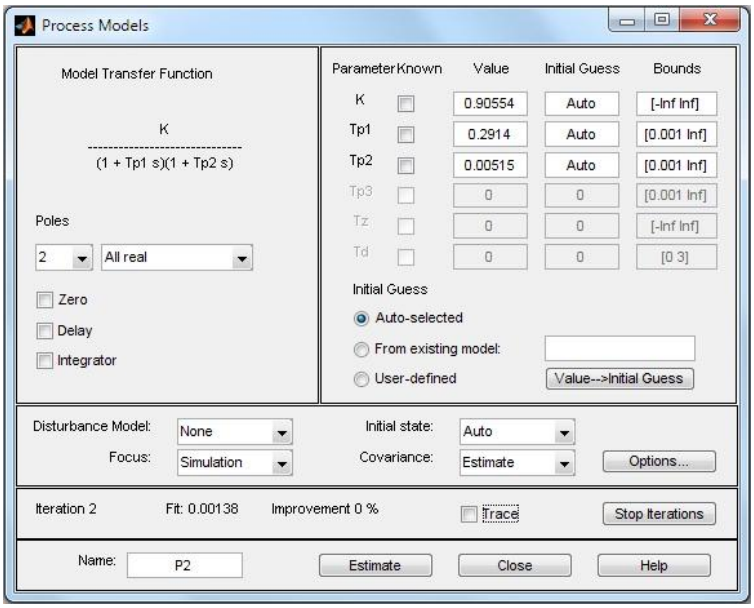
5.5.2 Identifikace soustavy v Matlabu

Pro identifikaci soustavy, tedy získání přenosu G_S byl použit nástroj *Identification tool* v Matlabu, který se spouští příkazem *ident*. V něm pomocí nabídky *Time domain data…* jsou přidány vstupní a výstupní pole naměřených hodnot. Vstupní pole hodnot jsou hodnoty požadovaných otáček, tedy 5000 otáček/min, výstupní jsou pak hodnoty odezvy, tedy aktuální otáčky motoru. Zbývá už jen zvolit Sampling interval 0,1 s. Okno nástroje identifikace nyní obsahuje importovaná data *step_motor* jak lze vidět na obrázku Obr. 12. Následuje samotná identifikace systému. Pomocí nabídky *Estimate…* → *Process models* se otevře okno *Process models*, které lze vidět na obrázku Obr. 13.

Pro identifikaci bylo použito aproximace soustavou druhého řádu se zesílením K a časovými konstantami T_{p1} a T_{p2}. Po stisku tlačítka *Estimate* se provede výpočet těchto konstant a dostáváme operátorový přenos soustavy $G_S(s)$.



Obr. 12 Data *step_motor* připravena pro identifikaci

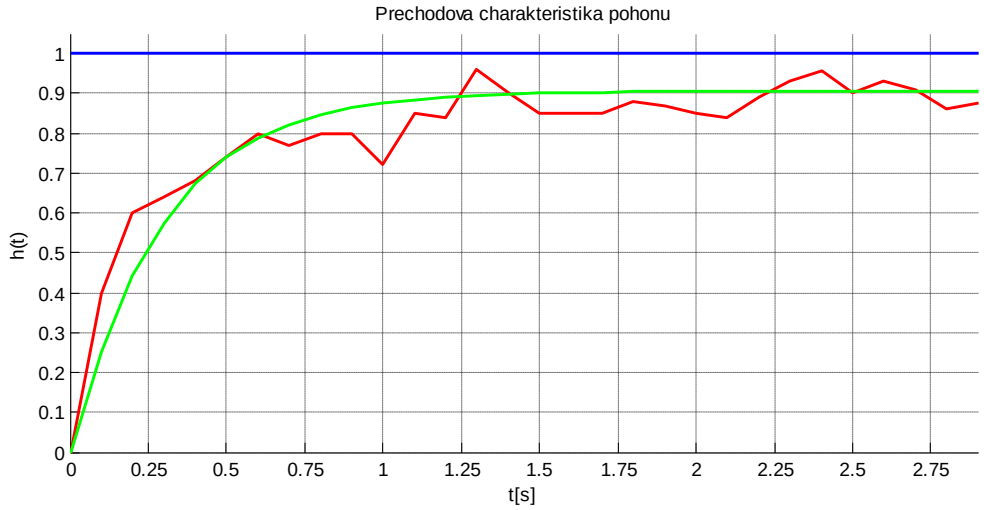


Obr. 13 Identifikace soustavy

Operátorový přenos soustavy je tedy:

$$G_S(s) = \frac{K}{(1+T_{p1}s)(1+T_{p2}s)} = \frac{0.90554}{(1+0.2914s)(1+0.00515s)} = \frac{0.90554}{\underline{0.0015s^2 + 0.2965s + 1}} \quad (11)$$

Na obrázku Obr. 14 je porovnání výsledků jednotkového skoku přivedeného na soustavu $G_S(s)$ (zelený průběh) se skutečnými naměřenými hodnotami (červený průběh). Přesnost aproximace $G_S(s)$ je postačující.



Obr. 14 Porovnání výsledků aproximace s naměřenými hodnotami

5.5.3 Diskretizace přenosu soustavy G_S

Jelikož bude použit diskrétní regulátor, je potřeba převést do diskrétní časové oblasti i přenos soustavy G_S. K tomu poslouží opět program Matlab a příkaz *c2dm* realizující převod spojitého systému do diskrétního času. Příkaz vrací koeficienty polynomu čitatele a jmenovatele diskrétního přenosu jako *numd* a *dend*. Při diskretizaci se zadává vzorkovací perioda T_v. Ta zároveň určuje, jak často bude prováděn zásah diskrétního regulátoru, to znamená, jak často se bude aktualizovat akční veličina regulátoru. Byla zvolena hodnota $T_v = 0.01\,s$.

```
[Gs_num_d, Gs_den_d] = c2dm([0.9055],[0.0015 0.2965 1.0000], 0.01, 'zoh')
```

Z koeficientů v maticích *Gs_num_d* a *Gs_den_d* pomocí příkazu *tf* dostaneme přenosovou funkci diskrétního systému. Rovněž zde byla použita zvolená vzorkovací perioda $T_v = 0.01\,s$.

```
Gs_d = tf(Gs_num_d, Gs_den_d, 0.01)
```

Výsledný přenos soustavy v diskrétním čase $G_S(z)$ (12).

$$G_S(z) = \frac{0{,}01714z + 0{,}009023}{z^2 - 1{,}11z + 0{,}1386} \tag{12}$$

5.5.4 Vytvoření PSD algoritmu v jazyce C

Po předchozích krocích následuje návrh vlastního PSD regulátoru. Protože není známa vnitřní struktura diskrétního PID regulátoru v Matlabu Simulinku, daleko přesnější metodou proto bude navrhnout regulátor v jazyce C, který bude později implementován v mikrokontroléru, a ten použít pro následnou simulaci a návrh konstant regulátoru. Tím se dostane daleko přesnější

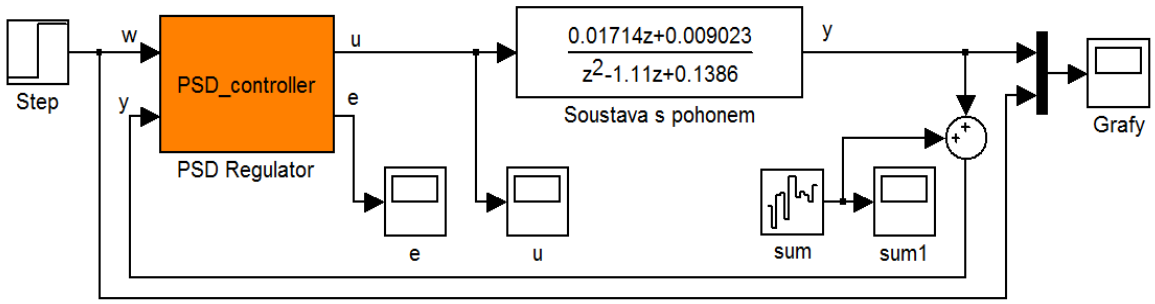
simulace realizovaného regulátoru a jeho chování, než v případě použití neznámého bloku PSD regulátoru Simulinku.

Tab. 1 Výpis programu – PSD algoritmus v jazyce C

```
// ********************** PID regulator ************************
//*************************************************************
float PIDLoopControl(PIDstruct *PSD){
       float pPart;
       float iPart;
       float dPart;
       float Ki;
       float Kd;

       // vypocet Ki, Kd
       Ki = (PSD->Kp * PSD->Ts)/PSD->Ti;
       Kd = (PSD->Kp * PSD->Td)/PSD->Ts;

       // vypocet regulacni odchylky
       PSD->error = PSD->setpoint - PSD->output;

       // vypocet PROPORCIONALNI slozky
       pPart = PSD->Kp * PSD->error;

       // vypocet SUMACNI slozky  (s antiwindupem)
       PSD->iState += PSD->error;
       iPart = Ki * PSD->iState;

      // Dynamicky antiwindup
       if(iPart > PSD->iMax){          // > max
              PSD->iState -= PSD->error;
              iPart = Ki * PSD->iState;
       }
       else if (iPart < PSD->iMin){  // > min
              PSD->iState -= PSD->error;
              iPart = Ki * PSD->iState;
       }

       // vypocet DERIVACNI slozky
       dPart = Kd * (PSD->error - PSD->dState);
       PSD->dState = PSD->error;

       // vypocet akcni hodnoty
       PSD->action = pPart + iPart + dPart;

       // limitace
      if(PSD->action > PSD->iMax) PSD->action = PSD->iMax;
      else if(PSD->action < PSD->iMin) PSD->action = PSD->iMin;

      // vraci hodnotu akcni veliciny
      return (PSD->action);
}
```

Algoritmus PSD regulátoru řeší i tzv. windup efekt. Ten nastává u diskrétních regulátorů tehdy, nemůže-li z jakéhokoliv důvodu výstup *y* dosáhnout požadované hodnoty *w*, tedy pokud je v ustáleném stavu regulační odchylka nenulová. V ten moment by rostl výsledek sumačního členu nade všechny meze. Aby se tomuto zabránilo, byl použit tzv. dynamický antiwindup. Sumační složka je počítána rekurzivně, a pokud vypočtená hodnota akční veličiny leží mimo realizovatelný rozsah, je růst integrační složky zastaven na předchozí hodnotě.

Uvedený PSD algoritmus v tabulce Tab. 1 byl přeložen kompilátorem pomocí funkce *mex* v Matlabu tak, aby se dal načíst v prostředí Simulink do bloku *PSD_controller* jako S-funkci. Schéma zapojení bloků v Simulinku je vidět na následujícím obrázku Obr. 15.



Obr. 15 Model v Simulinku

Do bloku PSD regulátoru vstupují zvlášť signály pro požadovanou hodnotu *w* a pro aktuální hodnotu výstupu *y*. Výstupem je pak akční veličina *u* působící na diskrétní soustavu s pohonem a pro informaci i regulační odchylka *e*. U všech bloků je nutné nastavit vzorkovací periodu $T_v = 0{,}01\,s$.

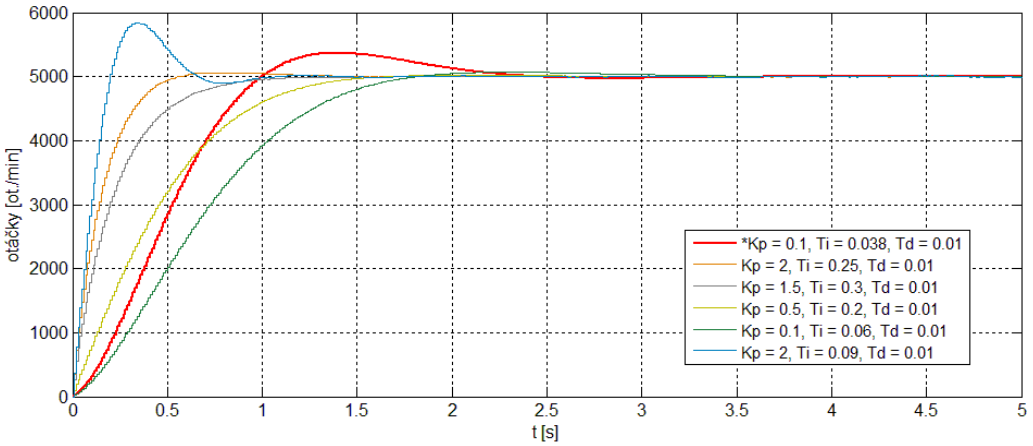
Na vstup je opět přiveden jednotkový skok. Na výstupu soustavy s pohonem je aktuální hodnota otáček *y*. K těm je přidán náhodně generovaný šum, simulující kolísání snímaných otáček, který se bude pohybovat v rozmezí ± 70 otáček/min.

5.5.5 Ladění konstant PSD regulátoru

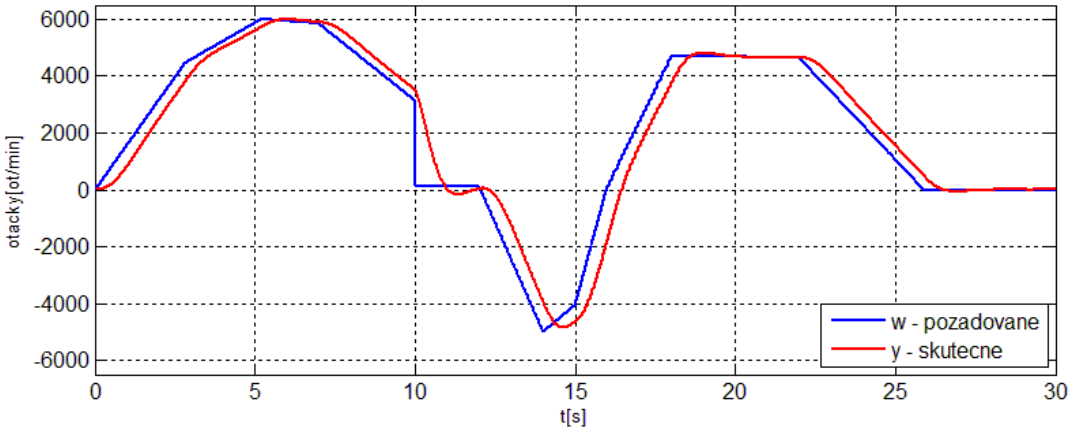
Pro nalezení konstant regulátoru bylo zamýšleno zvolit metodu nalezení konstant pomocí nástroje *sisotool* v Matlabu. Tyto konstanty však ve spojení s vlastním vytvořeným regulátorem v Simulinku byly nepoužitelné. Proto byla nakonec použita experimentální metoda. Postupné zadávání hodnot do proměnných K_p, T_i, T_d a sledování grafů s odezvou na jednotkový skok vedlo k nalezení optimálních hodnot, při kterých došlo ke splnění následujících požadavků:

- **optimální doba regulačního děje** – Čím kratší je regulační děj, tím větší má soustava motoru špičkový proudový odběr a naopak. Zároveň nesmí být doba regulace příliš dlouhá.
- **odolnost proti kolísání výstupní veličiny –** Obecně platí, že regulátory s dominantní integrační složkou jsou „odolnější“ vůči šumu, neboť nestačí tak rychlé změny sledovat.
- **povolený překmit max. 10 % -** Průběh odezvy při skokové změně otáček na hodnotu 5000 ot/min tedy nesmí překročit hodnotu 5500 ot/min.

Na obrázku Obr. 16 jsou uvedeny grafy odezev soustavy pro různé hodnoty K_p, T_i a T_d. Ze všech testovaných konstant byly nakonec na základě testování na vozidle vybrány ty, jejichž odezva soustavy je zakreslena v grafu červenou barvou a hodnoty v legendě zvýrazněny hvězdičkou.



Obr. 16 Graf – Odezvy soustavy pro různé K_p, T_i, T_d



Obr. 17 Graf – Reakce výstupu systému na zadaný průběh požadovaných otáček

Navrhnutý regulátor nepatří svou dobou regulačního děje mezi nejrychlejší, ale navrhnuté řešení spojuje všechny požadavky kladené na regulaci pohonu v této aplikaci. Pro názornost je na obrázku Obr. 17 zobrazena reakce výstupu systému na zadaný průběh požadovaných otáček.

6 Správa napájení a dobíjení zařízení

Hlavním úkolem správy napájení a dobíjení neboli správy energie je zaručit distribuci elektrické energie pro elektroniku průzkumného vozidla. Jeho úkolem tedy bude nabíjet, dobíjet a sledovat stav akumulátoru a ten dále vyhodnocovat. Pro komunikaci s nadřazenými a podřazenými moduly bude vybaven komunikačními periferiemi.

Základním kritériem pro návrh modulu správy energie je výběr vhodného akumulátoru, který nám omezuje jak maximální nabíjecí a vybíjecí proud, tak také svou kapacitou i dobu chodu celého zařízení.

6.1 Výběr zdroje energie

Na trhu existuje mnoho typů akumulátorů. Každý z nich má své výhody i nevýhody. Hlavními kritérii pro výběr akumulátoru jsou hustota energie ve Wh/kg, počet nabíjecích cyklů, životnost a v neposlední řadě jeho cena. Nejpoužívanější typy akumulátorů jsou spolu s jejich parametry shrnuty v následující tabulce Tab. 2.

Tab. 2 Vlastnosti různých typů akumulátorů [14] [15] [16] [17] [18] [19]

Typ akumulátoru	Jmenovité napětí [V]	Hustota energie [Wh/kg]	Samovybíjení [%/měsíc]	Počet nabíjecích cyklů
NiCd	1,2	40 ÷ 60	10	2000
NiMH	1,2	30 ÷ 80	30 (závislé na teplotě okolí)	500 ÷ 1000
SLA - (olověný akumulátor)	2.105	30 ÷ 40	3 ÷ 30	500 ÷ 800
Li-ion	3,6 / 3,7	100 ÷ 160	8 ÷ 31 (závislé na teplotě okolí)	1200
Li-pol	3,7	130 ÷ 200	5	>1000
LiFePo	3,3	90 ÷ 110	-	2000

Nejperspektivnějším akumulátorem elektrické energie v chemické podobě se jeví akumulátor typu Lithium – Polymer. Jejich vynikající vlastnosti, jako velmi nízká hmotnost, rozměry, vysoké proudy, vysoká životnost a velké napětí je přímo předurčují do oblasti modelářských pohonů a aplikací. Pokud se dodržují alespoň základní zásady bezpečnosti provozu, nejsou nebezpečné. Jednotlivé články akumulátoru lze řadit sérioparalelně a tím vytvářet různé kombinace o velikostech napětí a kapacit. Označení akumulátoru může obsahovat informaci o počtu článků a jejich řazení, např. 2s2p – 2 paralelní větve, každá tvořena dvěma články v sérii. Základní vlastnosti Li-pol akumulátorů:

- Výhodný poměr váha / kapacita (130–200 Wh/kg) v porovnání s ostatními typy akumulátorů, Minimální samovybíjení (procenta za měsíc)
- Jmenovité napětí jednoho článku 3,6 V (některé prameny uvádějí také 3,7 V)
- Maximální vybíjecí proud až 20 C, nabíjecí proud 1 C, maximální napětí článku 4,2 V

– Nemá paměťovou depresi či „paměťový jev“, články není třeba před nabíjením vybíjet

Obr. 18 Akumulátor Li-pol

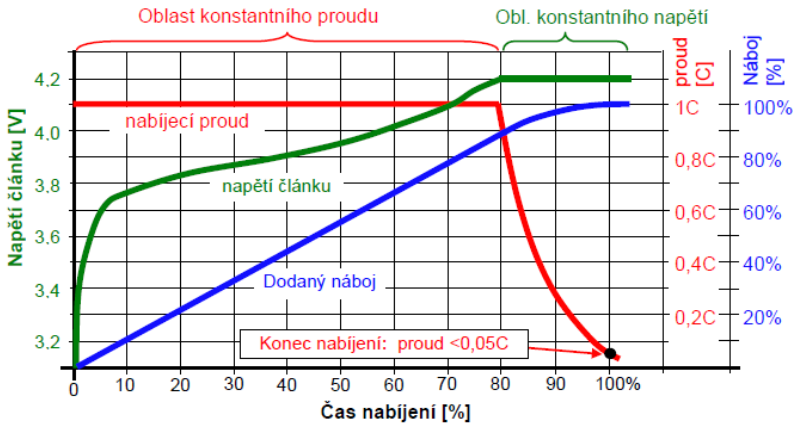
Akumulátory Li-pol obsahují kromě vývodů kladného a záporného pólu akumulátoru také tzv. servisní konektor. Na něm jsou vyvedena jednotlivá napětí z každého článku v akumulátoru. Počet pinů tohoto konektoru se liší podle počtu článků, např. u 6tičlánkového akumulátoru najdeme sedmi-pinový konektor. Používá se pro zjišťování napětí na jednotlivých článcích a jejich vyvažování na stejnou napěťovou úroveň. O jeho funkci více v kapitole o *6.2.1 Balancování článků*.

Jako akumulátor byl na základě uvedených vlastností vybrán modelářský Li-pol akumulátor firmy Polyquest nesoucí označení PQ-4350XP. Jedná se o akumulátor na obrázku Obr. 18 s následujícími vlastnostmi:

– Jmenovité napětí 22,2 V při řazení článků 6s1p
– Kapacita akumulátoru 4350 mAh; maximální vybíjecí proud 20 C, tedy 87 A
– Rozměry (v mm): 166 x 46 x 46, hmotnost 655 g, [12]

6.2 Nabíjení a dobíjení akumulátoru

Při nabíjení a dobíjení akumulátoru dochází obecně k akumulaci elektrické energie, která se převádí z elektrické podoby do formy chemické. V průzkumném vozidle byl použit Li-pol akumulátor. Zde je popsán princip nabíjení akumulátoru a jeho tzv. balancování.



Obr. 19 Průběh nabíjení Li-pol akumulátoru metodou CC – CV [9]

Li-pol/Li-Pol akumulátor se oproti NiCd nebo NiMH akumulátorům nabíjí podstatně lépe, neboť u něj nevzniká problém s detekcí ukončení nabíjení. S výhodou se pro něj používá metoda CC – CV (Constant Current – Constant Voltage) neboli konstantní napětí – konstantní proud, která zaručí nabití článku na jeho 100% kapacitu.

Pro práci s těmito akumulátory při nabíjení a vybíjení musí být splněny následující podmínky:

- Nabíjecí napětí nesmí přesáhnout 4,2 V ± 0,03 V / článek, nabíjecí proud 1 C max
- Teplota nabíjených článků v rozmezí 0 °C až 45 °C
- Minimální napětí, při kterém je nutno ukončit vybíjení je 3 V

6.2.1 Balancování článků

Tímto pojmem je myšleno vyrovnávání napětí u jednotlivých článků v akumulátorové baterii během jejich nabíjení. Protože se ne každý článek vybije na stejné napětí, stává se pak při jejich nabíjení, že články dosáhnou svého stavu nabití v navzájem různých časech. Takto nabité nevyvážené články se opět nestejnoměrně vybíjejí a celý problém se opakuje. Aby se tomuto zamezilo, používá se obvod zvaný balancer, který během nabíjení vyrovnává napětí na článcích tím, že příslušný článek s napětím vyšším než mají ostatní, začne vybíjet malým proudem tak dlouho, dokud jeho napětí neodpovídá ostatním článkům. Pokud by tedy nastala situace, že jeden z článků by měl menší napětí než ostatní, začaly by se tyto ostatní vybíjet malým proudem tak dlouho, dokud na všech článcích nebude stejné napětí.

Balancování se provádí pouze při nabíjení. Proto při provozu akumulátoru, kdy se nebalancuje, je nutno sledovat napětí na jednotlivých článcích, a pokud by na některém z článků bylo napětí nižší než 3 V, je nutné odběr proudu z akumulátoru zastavit. Platí totiž, že články s nižším svorkovým napětím než ostatní, se vybíjejí rychleji. To by mohlo vést k jejich znehodnocení (podbití).

Samotné nabíjení a dobíjení článků řešil jiný student v rámci své bakalářské práce. Proto je zde dále uvedeno sledování stavu akumulátoru, které je řešeno v této práci.

6.2.2 Sledování stavu akumulátoru

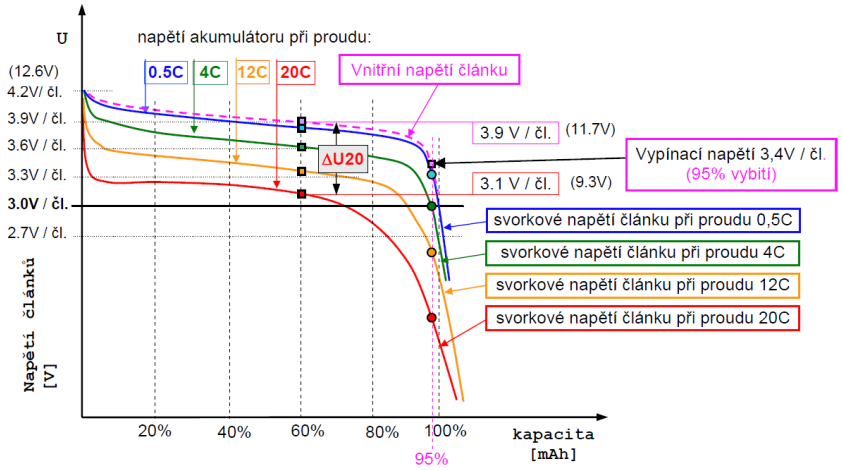
Během provozu (vybíjení) akumulátoru a potažmo celého průzkumného vozidla je potřeba sledovat stav akumulátoru. Mezi sledované parametry patří:

- míra nabití v procentech
- zbývající doba do vybití
- hlídání stavu napětí a proudu akumulátoru, ochrana proti podbití a nadproudu

Pro zjištění míry vybití akumulátoru je možné použít dvě metody. První z nich využívá přímou úměru mezi vnitřním (elektromotorickým) napětím akumulátoru a mírou jeho nabití. To má však jednu nevýhodu. Vnitřní napětí akumulátoru nelze měřit přímo. Měřit lze pouze svorkové napětí. Svorkové napětí je stejné jako vnitřní napětí pouze v jediném případě – v případě nulového odebíraného proudu. Ve všech ostatních případech je svorkové napětí menší o napětí ΔU_{xx}, které

se „zachytí“ na vnitřním odporu článku (R vnitřní) působením protékajícího proudu a je závislé jak na tomto odporu, tak na protékajícím proudu. Pokud ale známe velikost vnitřního odporu (lze jej zjistit opět pouze měřením jiných veličin a vypočítat, nelze měřit přímo) a známe velikost protékajícího proudu, můžeme i ze svorkového napětí vypočítat napětí vnitřní (13) – a dostáváme se na „vybíjecí křivku“ vnitřního napětí akumulátoru (Obr. 20).

$$U_{VNITRNI} = U_{SVORKOVE} + \left(R_{VNITRNI} \cdot I\right) \tag{13}$$



Obr. 20 Graf – Závislost vypínacího napětí na velikosti vybíjecího proudu [9]

Výše uvedená metoda je přesná, ale je nutné znát vnitřní odpor akumulátoru. Druhou metodou jak zjistit míru vybití akumulátoru, je porovnání skutečné dodané kapacity během nabíjení s kapacitou odebranou během vybíjení. Tato druhá metoda byla nakonec použita a podávala výsledky s přesností $\pm 2\,\%$.

Metoda vyžaduje opět měření proudu, a to jak vybíjecího, tak i nabíjecího. Numerickou integrací (sumací) nabíjecího a odebraného proudu pak dostáváme celkovou dodanou nebo odebranou kapacitu (14).

$$C = \sum_{k=1}^{n} \frac{i_k \cdot T_V}{3600}, \quad [mAh; mA, s] \tag{14}$$

kde

C - kapacita v mAh

i - aktuální hodnota proudu v mA

T_V - vzorkovací perioda

Pokud jsou známy tyto hodnoty kapacit, dá se podle vzorce (15) zjistit stav akumulátoru v procentech.

$$stav\_akumulatoru = \frac{C_{DODANA} - C_{ODEBRANA}}{C_{DODANA}} \cdot 100, \quad [\%; mAh, mAh, mAh] \qquad (15)$$

kde

$stav\_akumulatoru$ - zbývající kapacita akumulátoru vyjádřena v procentech

C_{DODANA} - dodaná kapacita při nabíjení v mAh

$C_{ODEBRANA}$ - odebraná kapacita v mAh

Dalším počítaným údajem je zbývající doba běhu akumulátoru v sekundách. Vypočte se dle vztahu (16).

$$T_{ZBYVAJICI} = T_{CELKOVA}\left(\frac{C_{DODANA}}{C_{ODEBRANA}} - 1\right), \quad [s; s, mAh, mAh] \qquad (16)$$

Aby bylo možné sledovat stav akumulátoru, je potřeba proměnné používané ve výpočtech uchovávat v EEPROM paměti mikrokontroléru. Mezi takové proměnné patří:

- Dodaná kapacita při nabíjení [mAh]
- Odebraná kapacita vybíjením [mAh]
- Doba vybíjení akumulátoru [s]
- Doba nabíjení akumulátoru [s]

Tyto proměnné jsou aktualizovány v paměti EEPROM každých 5 sekund.

Po dokončení nabití akumulátoru, hodnota dodané kapacity nabíjením C_{DODANA} se aktualizuje novou hodnotou, která byla naměřena během nabíjení. V ten samý okamžik se nuluje hodnota kapacity $C_{ODEBRANA}$, aby se připravila pro nové měření odebrané kapacity během používání (vybíjení) akumulátoru.

Kromě sledování stavu akumulátoru je potřeba ho také určitým způsobem ochránit. Nabízí se ochrana proti nadproudu a podpětí. Oba nežádoucí stavy můžou vést až ke zničení akumulátoru. Proto se aktuální napětí akumulátoru a odebíraný proud snímají a dále zpracovávají.

Ochrana proti nadproudu

Hodnota proudu nesmí překročit za žádných okolností nastavenou mez. Ta je nastavena s dostatečnou rezervou kvůli špičkovým odběrům na hodnotu 10 A. Pokud proud vzroste nad tuto mez, modul správy energie automaticky zastaví chod palubní elektroniky a znovuuvedení do chodu je možné pouze restartováním vozidla.

Nastavená mez je dostatečná a během testování nedošlo ani jednou k jejímu dosažení.

Ochrana proti podpětí

Velikost napětí baterie nesmí klesnout pod nastavenou mez a v ní setrvat minimální nastavený čas. Tento minimální čas je zde kvůli napěťovým poklesům, ke kterým dochází při velkých špičkových proudových odběrech. Tyto krátké napěťové poklesy by způsobovaly zbytečnou aktivaci ochrany podpětí. Minimální povolené napětí 6článkového akumulátoru je

$6 \times 3{,}3V = 19{,}8V$, proto byla ochrana nastavena na $20V$ s časovou prodlevou 5 sekund. Pokud tedy dojde k poklesu napětí pod 20 V na dobu delší než 5 sekund, je akumulátor považován za vybitý. To má za následek odpojení zbytku elektroniky vozidla a vyčkání obsluhy na připojení zdroje nabíjení.

6.3 Filtrace analogových veličin

Při použití analogových měření vyvstává problém se zašuměním užitečného napěťového signálu a dochází tak k jeho degradaci. V mnoha případech se proto používá různých filtrů, které pomáhají tyto nežádoucí vlivy potlačit.

V tomto případě, kde budou měřeny analogové veličiny a převáděny do diskrétní časové oblasti, lze vybírat ze spojitých nebo diskrétních filtrů. Realizace filtrů ve spojité časové oblasti je obvodově složitá, zvláště pak při implementaci filtrů vyšších řádů. Proto bude navrhnut filtr v diskrétní časové oblasti. Shrnutí výhod a nevýhod diskrétních a analogových filtrů lze vidět v následujícím přehledu [20] :

a) Číslicový filtr

- Vysoká přesnost
- Nemají drift
- Mohou mít lineární fázi
- Možnost adaptivní filtrace
- Snadná simulace a návrh
- Výpočet musí proběhnout během periody vzorkování
- Filtrace nízkých frekvencí

b) Analogový filtr

- Menší přesnost
- Drift vlivem změn parametrů součástek
- Nelineární fáze
- Adaptivní filtrace nejde
- Obtížná simulace a návrh
- Vhodné pro vysoké frekvence

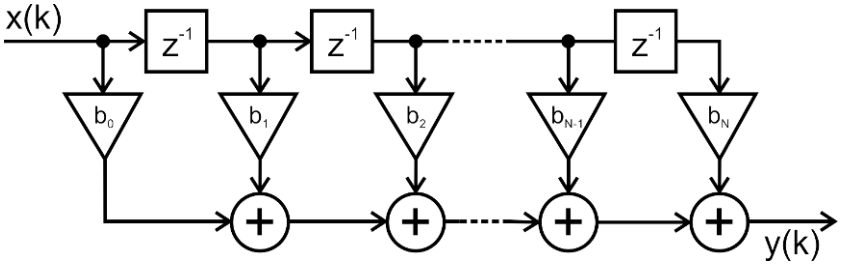
Jedním z hojně používaných filtrů je filtr FIR, se kterým je možné sestrojit tzv. plovoucí průměrování, což je výpočet aktuální hodnoty aritmetickým průměrem z několika posledních naměřených hodnot.

6.3.1 FIR filtr

Filtr s konečnou impulzní odezvou, zkráceně FIR, je diskrétní lineární filtr, který má konečnou impulzní odezvu. Protože je filtr nerekurzivní (tj. nemá žádné zpětné vazby), je vždy stabilní. Filtr má poměrně jednoduchý návrh. Výstup FIR filtru lze popsat vztahem (17).

$$y(k) = \sum_{i=0}^{N-1} b_i x(k-i) \tag{17}$$

Na obrázku Obr. 21 je vidět realizace FIR filtru, při kterém je potřeba bloků sčítačka, násobení konstantou a jednotkové zpoždění.



Obr. 21 Blokové schéma FIR filtru [20]

6.3.2 Plovoucí průměr a odhad plovoucího průměru

Pomocí FIR filtru lze jednoduše realizovat aritmetické průměrování z několika posledních naměřených hodnot, nazývané jako plovoucí průměrování, pokud se $b_i = \frac{1}{N}$. Potom lze vztah (17) přepsat na (18) a tím dostat vztah pro plovoucí průměrování.

$$y(k) = \frac{1}{N} \sum_{i=0}^{N-1} x(k-i) \tag{18}$$

Plovoucí průměr může být počítán rekurzivně (19) a tím se dá vyhnout výpočtu sumy posledních N měření.

$$y(k) = \frac{\left(y(k) \cdot N - x(k-(N-1)) + x(k)\right)}{N} \tag{19}$$

Ve vztahu (13) figuruje člen $y(k) \cdot N$ jako součet členů za posledních N měření. Při výpočtu aktuální hodnoty $y(k)$ je odečtena od součtu nejstarší hodnota $x(k-(N-1))$ a přičtena hodnota aktuální $x(k)$. Vydělením hodnotou N je získán aritmetický průměr z posledních N hodnot. Tato metoda filtrace sice nemusí v každém kroku vypočítávat sumu z posledních n měření, musí si ale přesto pamatovat hodnoty z posledních *N* měření.

O mnoho jednodušší a rychlejší metodou je odhad plovoucího průměru. Zakládá se na tom, že místo odečtení hodnoty nejstaršího měření $x(k-(N-1))$ se odečte samotná průměrovaná hodnota $y(k)$. Hodnota $y(k)$ je tedy odhadem hodnoty nejstaršího měření. Tato metoda umožňuje provádět filtraci rychle a bez potřeb pamatovat si *N* posledních naměřených hodnot. [21]

$$y(k) = y(k) + \frac{\left(x(k) - y(k)\right)}{N} \tag{20}$$

Pro filtraci byla nakonec použita poslední popisovaná metoda filtrace, tedy odhad plovoucího průměru dle vzorce (20).

7 Hardwarový návrh modulů

Pro navrhovanou aplikaci bylo potřeba navrhnout schémata zapojení a předlohy pro výrobu desek plošných spojů. Schéma zapojení bylo navrženo v programu Formica Schematic Editor v4.40, předlohy pro výrobu DPS pak v programu Formica Layout Editor v4.40.

Obě desky plošných spojů jsou koncipovány jako oboustranné s kombinovanou metodou osazení, tedy SMT i Through-hole technologií.

Schémata zapojení a masky pro výrobu plošných spojů jsou uvedeny v přílohách.

7.1 Výběr vhodné platformy

Díky dosavadním zkušenostem s mikrokontroléry Freescale a jejich zaměřením na automobilový průmysl byla následně vybrána platforma HCS12.

7.1.1 Základní vlastnosti

Pro řízení modulů bylo v obou případech použito mikrokontroléru Freescale MC9S12DP512. Ten v sobě zahrnuje velké množství periferií pro komunikaci a sběr dat z okolí.

MC9S12DP512 je 16bitový mikrokontrolér skládající se ze standardních integrovaných periferií, které jsou uvedeny v následujícím přehledu. [5]

- 16bitová šířka sběrnice
- rychlost sběrnice až 25 MHz
- 512 kB Flash EEPROM paměti pro program
- 14 kB RAM paměti pro data
- 4 kB EEPROM pro data
- 2x sériové komunikační rozhraní SCI
- 3x sériové periferní rozhraní SPI
- osmikanálový rozšířený záchytný systém / časovač
- 2x osmikanálový 10bitový AD převodník (ADC)
- osmikanálová pulzně šířková modulace (PWM)
- sběrnice I2C
- 5x modul CAN 2.0 A, B s maximální přenosovou rychlostí 1 Mbps

7.2 Modul pohonu

Modul pohonu bude regulovat otáčky pohonu, číst data z enkodéru, sledovat velikost odebíraného proudu a dále tyto data vyhodnocovat. Jako základní řídicí prvek bude použito mikrokontroléru Freescale řady HCS12. Modul bude obsahovat sběrnici CAN pro vzájemnou komunikaci s dalšími zařízeními.

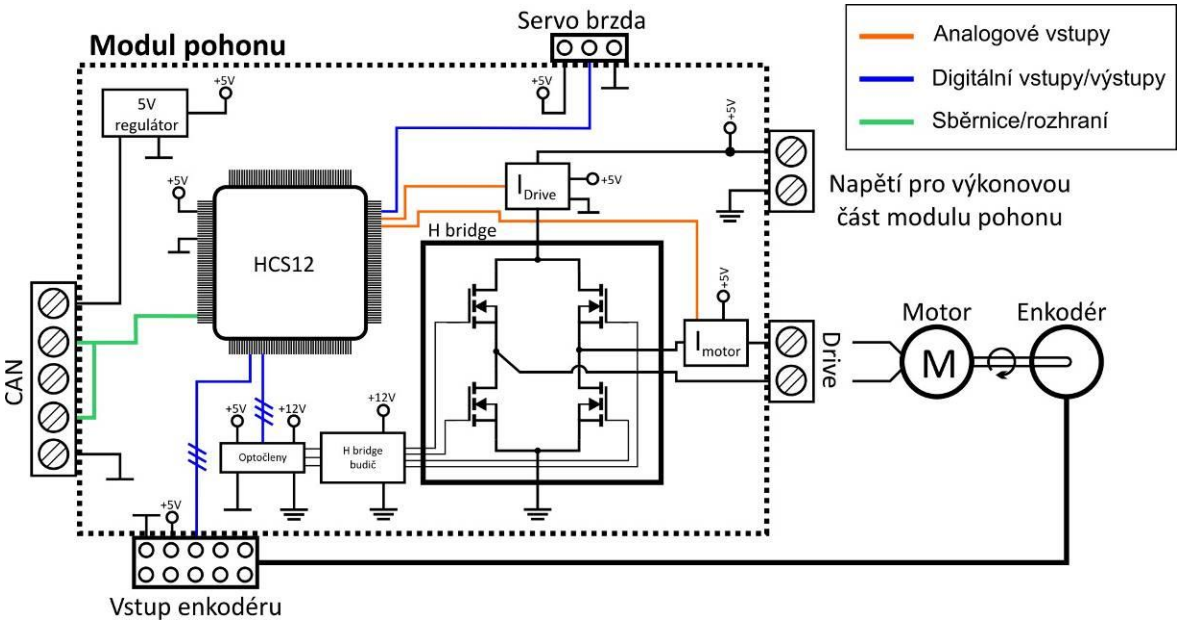
7.2.1 Podrobné blokové schéma

Podrobné požadavky a funkce modulu správy energie:

- Výkonový stupeň pro připojení pohonu – H-bridge
- Připojení kvadraturního enkodéru se signály A, B, I
- Měření proudu na výkonovém stupni a na motoru
- Servokanál pro ovládání mechanické brzdy
- Oddělení signálové a silové země
- Řízení modulu po sběrnici CAN

Údaje pro dimenzování a výběr komponent:

- Maximální proud motoru: 20 A
- Maximální napájecí napětí motoru: 30 V
- Maximální frekvence vstupu enkodéru 50 kHz
- Napájení signálové části: 14 V



Obr. 22 Blokové schéma modulu pohonu

7.2.2 Popis zapojení

Na obrázku Obr. 22 lze vidět podrobné blokové schéma modulu pohonu. Schéma je rozděleno na část silovou a část signálovou, přičemž je dodrženo oddělení silové země od signálové a jejich možnost následného spojení v jednom bodě.

Signálová část je napájena napětím 5 V ze spínaného regulátoru LM2595. Vstupní napětí regulátoru o hodnotě 14 V je přivedené na konektor sběrnice *CAN*.

Silová část je napájena nestabilizovaným napětím o velikosti 22 V, které je k dispozici na konektoru *Drive power supply*.

Signálová část se skládá ze zapojení mikrokontroléru HCS12, transceiveru CAN, měření proudu na bázi Hallova jevu, napájení vstupů optočlenů a také napájení enkodéru motoru, jehož 10pinový konektor je na desce pod názvem *Encoder input*. Do signálové části bylo také zahrnuto servo, hlavně z důvodu jeho 5V napájení.

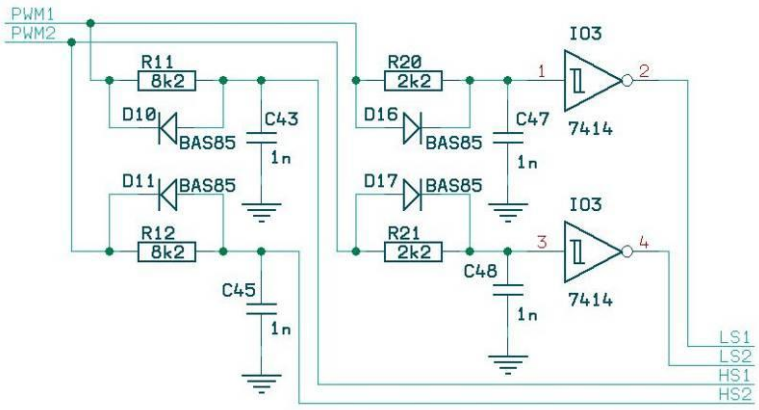
Silová část se skládá ze zapojení výstupů optočlenů, obvodu MC33883 pro buzení MOSFET tranzistorů H-bridge a samotného H-bridge obsahující tranzistory N MOSFET IRF2804.

Na desce modulu jsou přidány LED diody pro zobrazení stavu zařízení a je zde také vyveden konektor rozhraní BDM pro programování mikrokontroléru.

a) Měnič

Komplementární signál PWM z mikrokontroléru je spolu s povolovacím signálem *enable* opticky oddělen od silové části a je přiveden na vstup budiče MC33883. Ten zaručuje spolehlivé a rychlé sepnutí tranzistorů H-bridge.

Jako spínací tranzistory byly použity N MOSFET IRF2804. Ty vynikají svým malým odporem v sepnutém stavu $R_{DS(on)} = 2m\Omega$ a z toho také velkou proudovou zatížitelností až $75\,A$.



Obr. 23 Realizace zpoždění vzestupných a sestupných hran PWM signálu

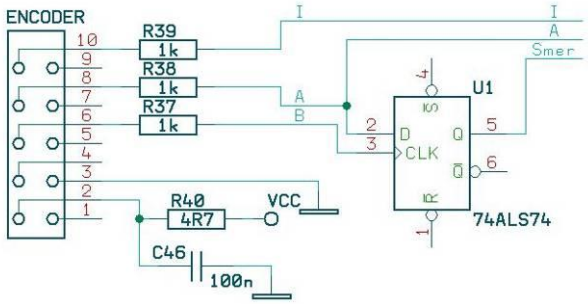
U měniče je dle obrázku Obr. 23 vhodným zapojením RC filtrů s diodou a Schmittových invertorů také vyřešen tzv. mrtvý čas neboli *dead band*. Jedná se o dobu, mezi uzavřením horního tranzistoru a otevřením dolního tranzistoru a naopak. Zavedením tohoto malého zpoždění (řádově

µs) posouvajícího hrany komplementárního PWM signálu dochází k odstranění nežádoucích tepelných ztrát v okamžiku otevírání a uzavírání tranzistoru.

Měnič dále obsahuje dvě měření proudu, do motoru a do celého měniče. Byly zvoleny převodníky ACS712 s Hallovou sondou a výstupem 0 ÷ 5 V odpovídající rozsahu proudu ±20 A.

b) Enkodér

Konektor pro enkodér obsahuje trojici digitálních signálů (A, B, I), napájecí napětí 5V pro elektroniku enkodéru a signálovou zem. Pro zjištění směru otáčení pohonu je na desce modulu pohonu umístěn vhodně zapojený D klopný obvod využívající signály A, B. Jeho výstupem je pak hodnota logické 1 pro jízdu vpřed, a hodnota logické 0 pro jízdu vzad.



Obr. 24 Konektor enkodéru s klopným obvodem typu D

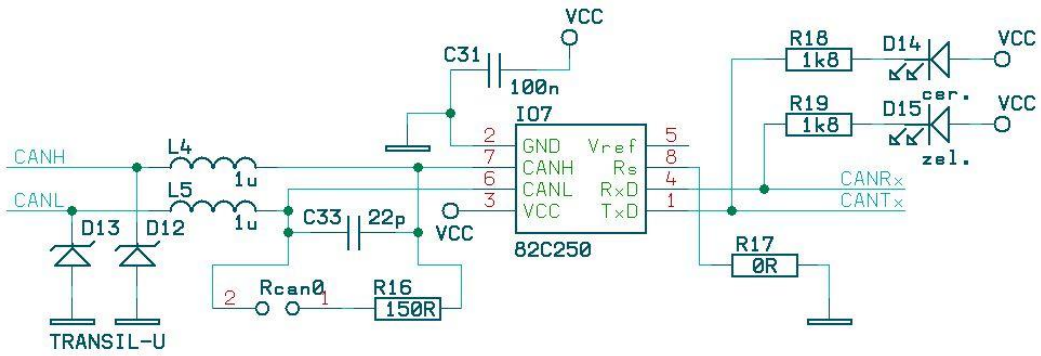
Zapojení na obrázku Obr. 24 s klopným obvodem typu D velmi ulehčuje práci mikrokontroléru, který by jinak musel vyhodnocovat směr otáčení softwarově v přerušení záchytného systému.

c) Servo

Konektor serva obsahuje 5V napájení, signálovou zem a pwm signál pro polohování serva.

d) Sběrnice CAN

Schéma zapojení CAN rozhraní je převzato z vývojových modulů EvB HCS12. Zapojení obsahuje CAN transceiver s pasivními součástkami pro nastavení, ochranu a filtraci.



Obr. 25 Schéma zapojení CAN rozhraní

7.3 Modul správy energie

Modul správy energie bude řídit nabíjení, dobíjení a sledovat stav baterie při vybíjení a dále jej vyhodnocovat. Jako základní řídicí prvek bude použito mikrokontroléru Freescale řady HCS12. Modul bude obsahovat sběrnici CAN pro vzájemnou komunikaci s dalšími zařízeními.

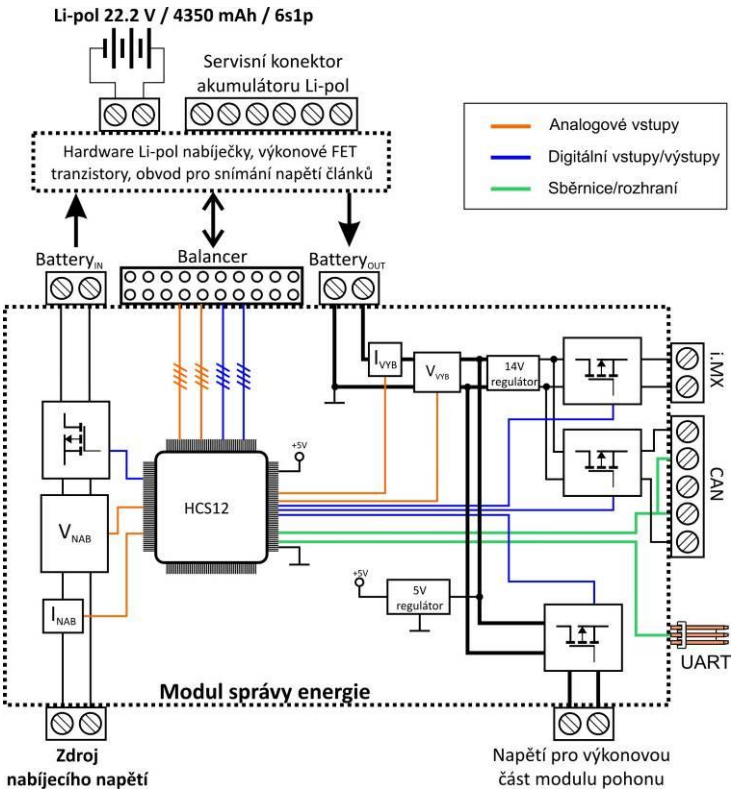
7.3.1 Podrobné blokové schéma

Podrobné požadavky a funkce modulu správy energie:

- Měření napětí akumulátoru a odebíraného proudu
- Měření nabíjecího napětí a proudu, řízení nabíjení akumulátoru
- Použití Li-pol akumulátoru
- Balancování Li-pol článků akumulátoru
- Distribuce 14V napájení pro ostatní moduly (spínání napájecích větví pro jednotlivé moduly)
- Řízení modulu po sběrnici CAN, 3V UART

Údaje pro dimenzování a výběr komponent:

- Maximální nabíjecí proud: 4 A
- Maximální nabíjecí napětí: 30 V
- Maximální proud z akumulátoru: 10 A
- Maximální napětí akumulátoru: 26V
- Maximální možný počet balancovaných Li-pol článků: 6 čl.
- Počet spínaných 14V napájecích větví: 3 větve / odběr dohromady max. 3 A



Obr. 26 Blokové schéma modulu správy energie

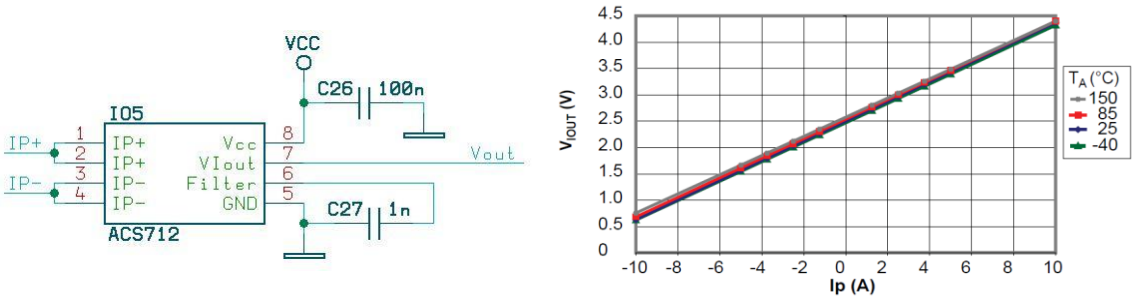
7.3.2 Popis zapojení

a) Měření proudu

Pro měření proudu označeného ve schématu jako I_{NAB} a I_{VYB} bylo použito snímačů na bázi Hallova principu. Ty zaručují minimální úbytek napětí při měření protékajícího proudu (odpadá použití bočníku při měření), což je ale zčásti vyváženo větší náchylností k šumu. Ta se dá do jisté míry snížit zmenšením šířky pásma signálu měřeného proudu – použitím filtru.

Byl vybrán Hallův senzor ACS712 s rozsahem ±20 A, jehož výstupem je analogová hodnota 0 až 5 V.

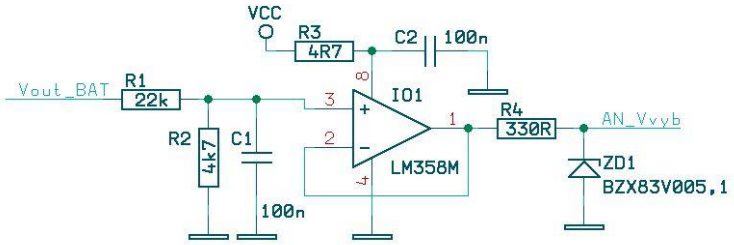
Zapojení Hallova senzoru na obrázku Obr. 27 ACS712 umožňuje připojit na vstup *Filter* filtrační kondenzátor a tím částečně potlačit šum výstupní analogové hodnoty. Velikost kapacity $1\,nF$ odpovídá dle katalogového listu zlomové frekvenci $45\,kHz$.



Obr. 27 Schéma zapojení a závislost výstupního napětí na hodnotě měřeného proudu

b) Měření napětí

Pro měření napětí označených ve schématu jako V_{NAB} a V_{VYB} je použito zapojení s operačním zesilovačem LM321. Na jeho vstupu je napěťový dělič, který převádí vstupní napětí v rozsahu 0 až 28 V na rozsah 0 až 5 V. Na výstupu OZ je pak klasické zapojení se Zenerovou diodou mající ochránit analogový vstup mikrokontroléru před přepětím. Nechybí zde ani filtrace napájecího napětí.



Obr. 28 Schéma zapojení měření napětí

c) Balancování článků

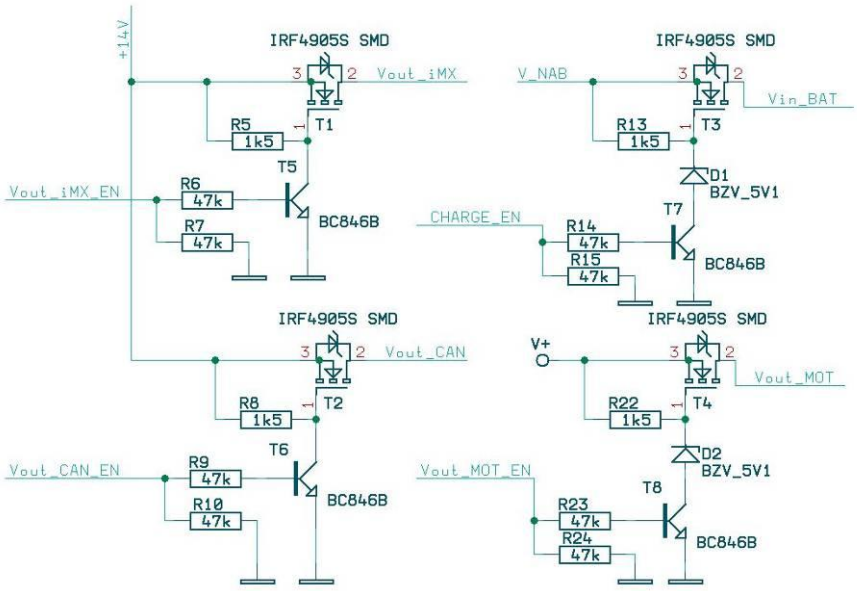
Pro řízení balancování a nabíjení článků bude využito mikrokontroléru, nicméně výkonová část nabíjení a balancování bude umístěna na externí desce plošných spojů a desky navzájem propojeny balancovacím konektorem obsahujícím tyto signály:

- 7x analogový signál o velikosti napětí na jednotlivých článcích
- 7x digitální (PWM) signál jako akční hodnota ovlivňující průběh balancování článků

Na externí desce, kterou souběžně řešil jiný student v rámci své bakalářské práce, je umístěna šestice diferenčních operačních zesilovačů měřících jednotlivá napětí na článcích. Dále šestice MOSFET tranzistorů, které budou podle potřeby balancování připínat výkonové odpory pro vybíjení (balancování) Li-pol článků.

d) Připínání napájecích větví

Modul správy energie má možnost podle stavu baterie a dalších požadavků odpojit napájecí větve pro další moduly. Děje se tak prostřednictvím P MOSFET tranzistorů s nízkou hodnotou odporu v sepnutém stavu $R_{DS(on)}$. Ta se u vybraných tranzistorů IRF4905S pohybuje okolo hodnoty $20\ m\Omega$.



Obr. 29 Spínané distribuční větve

Modul má možnost odpojit všechny podřízené uzly na sběrnici CAN, jakožto i nadřazený systém i.Mx. Oba jmenované celky jsou napájeny ze spínaného regulátoru o výstupním napětí 14 V a maximálním proudu 3 A. Dále má modul možnost odpojit napájení motoru, které je odebíráno přímo z akumulátoru.

Na obrázku Obr. 29 lze vidět čtveřici MOSFET tranzistorů. Prvním z nich je tranzistor *T3* sloužící pro připojení nabíjecího napětí pro akumulátor. Spínaní se provádí přes bipolární tranzistor *T7* signálem *CHARGE_EN*. Za povšimnutí stojí zapojení Zenerovy diody D1. Nabíjecí napětí přesahuje hranici 20 V, což je maximální možné napětí V_{GS}, kterým lze *T3* ovládat. Aby nedošlo k poškození tranzistoru, musí zde být použita Zenerova dioda s napětím alespoň 5,1 V.

Připojení 14V rozvodu pro CAN a i.Mx se děje pomocí tranzistorů *T2* a *T1* a ovládacích signálů *Vout_CAN_EN* a *Vout_iMX_EN*.

Pro spínání napětí pro motor slouží tranzistor *T4* a signál *Vout_MOT_EN*.

8 Operační systém

Existují dvě řešení jak navrhnout softwarovou strukturu požadovaného zařízení. Tedy s operačním systémem, nebo bez něj. Implementace zařízení bez operačního systému má své výhody, jako například cena, nebo menší náročnost na výpočetní výkon, ale u složitějších úloh jako je tato, převažují jeho nevýhody. To může být například nízká variabilita a špatná přenositelnost kódu.

Často navíc sama řízená úloha přímo vybízí k použití více-vláknového zpracování, což je jasný signál pro použití operačního sytému. Ulehčíme si tak práci využitím předem připravených synchronizačních nástrojů pro synchronizaci úloh a hardwarových přerušení.

Shrneme-li předchozí, využití operačního systému dává smysl u složitějších úloh, kde předpokládáme častou modifikaci programu (například přidání dalších senzorů), a proto je u obou modulů využit operační systém.

Jednotlivé řešení a jeho výhody jsou porovnány v následující tabulce Tab. 3.

Tab. 3 Porovnání struktur software

Bez operačního systému	S operačním systémem
+ jednoduchost	+ variabilita
+ cena vývoje	+ přenositelnost kódu
- přenositelnost kódu	+ synchronizační nástroje
	- větší nároky na výpočetní výkon
	- větší nároky na velikost paměti

8.1 Výběr vhodného operačního systému

Existuje celá řada dobrých operačních systémů (viz Tab. 4) určených, nebo alespoň částečně určených, pro malé vestavné (embedded) systémy, postavených na méně výkonných mikroprocesorech.

Po zvážení všech kladů a záporů operačních systémů různých výrobců, byl vybrán výrobce FreeRTOS a jeho stejnojmenný operační systém reálného času. Mezi jeho hlavní výhody patří [1] :

- Dostupné demonstrační aplikace pro různé architektury
- Je šířen pod GNU licencí (nemusí se publikovat zdrojové kódy)
- Je zdarma, volně stažitelný
- Nízké nároky na výpočetní výkon, dá se aplikovat i na 8bitové mikroprocesory.
- Podpora až 23 různých architektur

Tab. 4 Operační systémy různých výrobců

Výrobce	Název operačního systému
AVIX	AVIX-RT 2.2.1
CMX Systems	CMX 5.30
Express Logic	ThreadX G5.1.5.0
FreeRTOS	FreeRTOS v6.0.2
Micrium	μC/OS-II v2.84
Pumpkin	Salvo 4 Pro and Salvo 4 LE
RoweBots	DSPNanoUnison
Segger	embOS V3.52
OSEK	OSEK/VDX

8.2 FreeRTOS

FreeRTOS je operační systém reálného času pro vestavné zařízení. Téměř celý operační systém je napsán v programovacím jazyce C, pouze několik funkcí je vytvořeno v assembleru. Podporuje jak plně preemptivní tak kooperativní zpracován vláken. Jádro tohoto RTOS je složeno pouze ze tří souborů, a to *task.c, queue.c* a *list.c*. [1]

FreeRTOS je jednou ze tří distribucí, které výrobce poskytuje. Zbývající dvě verze jsou:

- **OpenRTOS** – komerční verze FreeRTOS, která sebou nese plnou podporu USB a TCP/IP stacku.
- **SafeRTOS** – certifikovaná verze pro bezpečnostní vestavné aplikace. Splňuje normu SIL3 dle IEC61508.

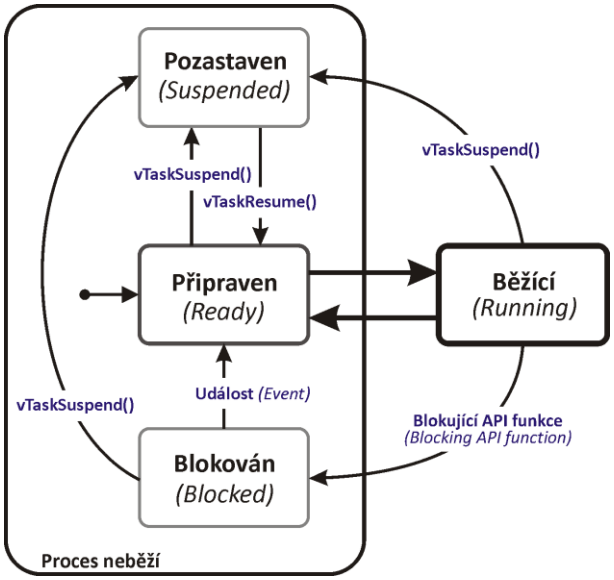
Pro komunikaci mezi vlákny, nebo mezi přerušeními a vlákny poskytuje komunikační a synchronizační nástroje typu *fronta zpráv, (binární) semafor a mutex*. Nástroj *mutex* zároveň řeší inverzi priorit. [2]

8.2.1 Práce s vlákny – plánovač

Srdcem každého operačního systému je plánovač (scheduler). FreeRTOS disponuje plánovačem s prioritním plánováním, kdy je každému vláknu přiřazena hodnota jeho priority. Menší číslo znamená nižší přiřazenou prioritu, přičemž nejnižší možná hodnota je 0. V případě vláken se stejnou prioritou jsou vlákna plánována metodou „kruhu“ (Round Robin). To znamená, že jsou pravidelně střídána. Maximální priorita je omezena na námi požadovanou hodnotu, kterou lze nastavit v souboru „FreeRTOSConfig.h“. [2]

Na následujícím obrázku Obr. 30 jsou znázorněny stavy, ve kterých se vlákno může nacházet. V jednom okamžiku může být procesoru přiděleno pouze jedno, a to odpovídá stavu „Běžící“. Ostatní vlákna jsou v tom okamžiku ve stavu „neběží“, který se pak dále rozděluje:

- **Pozastaven** (Suspended) – vlákna v tomto stavu nejsou přístupná plánovači. Každé vlákno lze umístit a vyjmout z tohoto stavu funkcemi *vTaskSuspend()* a *vTaskResume()*.
- **Připraven** (Ready) – vlákna v tomto stavu jsou připravena ke zpracování a čekají na přidělení procesorového času plánovačem.
- **Blokován** (Blocked) – vlákna čekající na událost. Události mohou být časové (vlákno je zablokováno na určitý čas) a synchronizační (semafor, mutex, fronta zpráv).



Obr. 30 Stavy, ve kterých se vlákno může nacházet [1]

8.3 Úprava FreeRTOS pro procesor MC9S12DP512

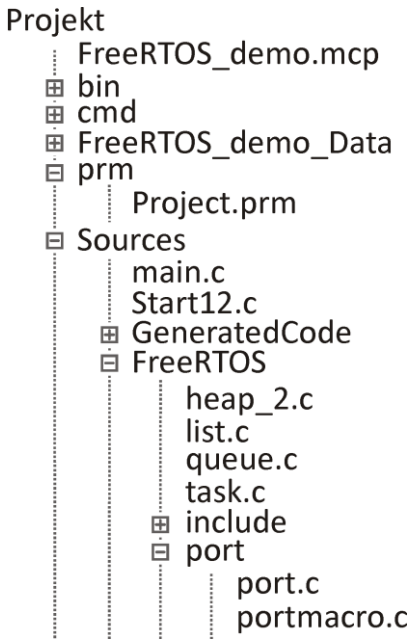
I když výrobce poskytuje mnoho předpřipravených projektů pro různé druhy mikroprocesorů, není v jeho silách obsáhnout všechny typy různých výrobců. Pro požadovaný mikroprocesor výrobce takzvaný port (projekt pro dané zařízení) neposkytuje, a proto bylo nutné ho vytvořit.

Nejprve bylo experimentováno s úpravou již existujícího projektu pro mikroprocesor MC9S12DP256B, na kterém bylo ověřeno, jak vlastně tento operační systém funguje, a co se musí zajistit pro jeho chod. Poté byl vytvořen nový projekt v prostředí CodeWarrior verze 5.9.0 a do něj přidány soubory nutné k běhu systému.

8.3.1 Vytvoření nového projektu

Po vytvoření nového projektu v prostředí CodeWarrior pro požadovaný typ procesoru, bylo nutné přidat soubory operačního systému a vytvořit jeho stromovou strukturu. Výrobce poskytuje operační systém jako celek, ve kterém se nacházejí jak jeho zdrojové kódy, tak všechny dostupné „porty“ pro různé typy mikroprocesorů. Tento nepřehledný balík dat se ale nehodí pro vyvíjení

aplikace, a proto byly vyjmuty pouze nutné soubory a vytvořena struktura, která je znázorněna na následujícím obrázku Obr. 31.



Obr. 31 Struktura projektu s operačním systémem

Popis důležitých položek:

- **Project.prm** – mapování paměti procesoru.
- **Sources** – složka, ve které jsou uloženy zdrojové kódy.
- **main.c** – v tomto souboru se nachází hlavní část uživatelského programu jako inicializace periferních zařízení, vytváření vláken a globálních synchronizačních nástrojů.
- **GeneratedCode** – tato složka obsahuje soubory, které definují takzvané „low level" funkce. Ty se provádí při spouštění procesoru.
- **FreeRTOS** – tato složka obsahuje soubory operačního systému.
- **list.c, queue.c, task.c** – soubory tvořící jádro operačního systému.
- **port.c, portmacro.c** – tyto dva soubory tvoří takzvaný „Port" operačního systému. Jsou to funkce, které jsou různé pro každou architekturu, a jsou potřeba pro chod systému.

8.3.2 Generování časových přerušení pro plánovač

Ke generování časových přerušení pro plánování vláken (scheduler) je využito sedmého kanálu mikroprocesorové periférie ECT (Enhanced Capture Timer), neboli časovače.

Frekvence sběrnice mikroprocesoru je nastavena na 25 Mhz, kterou je zároveň i inkrementován 16bitový čítač/časovač, jehož již zmíněný sedmý kanál generuje přerušení. Z toho plynou i omezení pro rozsah nastavení možných frekvencí spouštění plánovače.

Frekvence spouštění plánovače se nastavuje v konfiguračním souboru „FreeRTOSConfig.h“ položkou „configTICK_RATE_HZ“. Zde se může frekvence nastavit v rozsahu 382 Hz až 10000 Hz.

Výpočet minimální hodnoty (382 Hz) dle vzorce (21):

$$f_{MIN} = \frac{f_{BUS}}{2^{16}} = \frac{25 \cdot 10^6}{2^{16}} \doteq \underline{\underline{382\ Hz}} \tag{21}$$

Maximální hodnota není omezena schopností časovače dosáhnout takovéhoto kmitočtu, ale časem potřebným k vykonání obsluhy přerušení, tedy přepnutí kontextu. Při zvolené hodnotě 10000 Hz se tedy bude přerušení vyvolávat každých 100 µs. Měřením bylo zjištěno, že přepnutí kontextu (pouze dvě vlákna aktivní) trvá v průměru 12 µs. Z toho vyplývá, že pro jednoduché aplikace je možné tento kmitočet teoreticky využít, ale pro složitější už se budeme blížit periodě přerušení a na samotnou aplikaci nezbude skoro žádný výpočetní čas.

Ke spuštění časovače dochází v okamžiku volání funkce „*vTaskStartScheduler()*“. Jak má operační systém časovač nastavit je definováno v souboru port.c funkcí „*prvSetupTimerInterrupt()*“. Výpis této funkce je zobrazen v následující tabulce Tab. 5. První tři instrukce slouží k nastavení módu časovače a jeho vstupní děličky kmitočtu. Následuje výpočet hodnoty *TempTC7*, která určuje periodu přerušení a je v něm i využívána. Nakonec se časovač nastaví pro první spuštění a vynuluje se jeho příznak přerušení.

Tab. 5 Výpis programu – funkce nastavující časovač pro plánování. [1]

```
static void prvSetupTimerInterrupt( void )
{
        TIE = TIE | 128;        //TC7 povolení přerušení
        TSCR2 = TSCR2 | 0;      //Prescaler = 0
        TIOS = TIOS | 128;      //Kanál TC7 nastavit do módu Output Compare

        TempTC7 = 25000000/configTICK_RATE_HZ;    //Výpočet požadované periody
                                                  //přerušení časovače

        TC7 = TempTC7;          //Nastavení této periody

        TFLG1 = 128;            //Reset příznaku přerušení pro kanál TC7
}
```

Obsluha přerušení našeho časovače se také nachází v souboru port.c pod názvem „*vPortTickInterrupt()*“. V ní dochází k přeplánovaní vláken, a jelikož se jedná o přerušení, musí být umístěna mimo stránkovanou oblast paměti programu. To je zajištěno instrukcí preprocesoru „#pragma CODE_SEG __NEAR_SEG NON_BANKED“ před samotnou funkcí. Jako první se ve funkci vypočítává a nastavuje počet inkrementů časovače pro další přerušení. Za ní následuje rozhodovací instrukce preprocesoru, která určuje typ nastaveného plánování (preemptivní nebo kooperativní multitasking). Při preemptivní typu plánování dojde k uložení kontextu právě prováděného vlákna, k inkrementaci počítadla přerušení, k vlastnímu přeplánování a nakonec

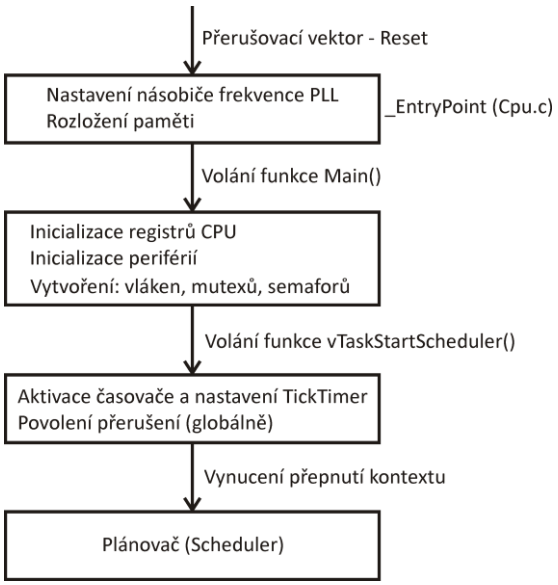
k obnově kontextu vlákna, které se bude zpracovávat. Pokud je využito kooperativního typu plánování, dochází pouze k inkrementaci počítadla přerušení.

Tab. 6 Výpis programu – obsluha přerušení časovače (plánování vláken) [1]

```
#pragma CODE_SEG __NEAR_SEG NON_BANKED
void interrupt vPortTickInterrupt( void )
{
        TC7 = TCNT + TempTC7;
        #if configUSE_PREEMPTION == 1
        {
                // Uložení kontextu, právě běžící úlohy
                portSAVE_CONTEXT();

                // Inkrementace počítadla ticku
                vTaskIncrementTick();

                // Přepnutí kontextu na jiné vlákno
                vTaskSwitchContext();

                // Obnovení kontextu nové úlohy
                portRESTORE_CONTEXT();
        }
        #else
        {
                vTaskIncrementTick();
        }
        #endif

        // Nulování příznaku přerušení časovače
        TFLG1 = 128;
}
#pragma CODE_SEG DEFAULT
```

8.3.3 Algoritmus spouštění operačního systému

Pro úplnost je uveden postup, jak vlastně dojde ke spuštění plánovače, a tím i operačního systému. Na obrázku Obr. 32 lze vidět algoritmus, který bude dále popsán.



Obr. 32 Spouštění operačního systému [1]

Po přivedení napájecího napětí k mikroprocesoru se program začne vykonávat od adresy vektoru přerušení definovaný pro reset. V tomto případě se nachází v souboru Cpu.c, ve kterém se provede nastavení frekvence sběrnice procesoru a inicializace paměti. Poté se již začne vykonávat uživatelská část programu, jako například vytvoření jednotlivých vláken, mutexů a front zpráv. V této chvíli uživatelem definovaná vlákna ještě neběží.

Voláním funkce „*vTaskStartScheduler()*“ mimo jiné dojde k povolení přerušení od periférií a k aktivaci časovače spouštějící plánovač. Nakonec se vyvolá softwarové přerušení, které zajistí první spuštění plánovače, ještě před jeho aktivací časovačem, dojde tedy k takzvanému vynucenému přepnutí kontextu. Při správném chodu by se program nikdy neměl vrátit zpět do funkce „*main()*“, ze které byl plánovač spuštěn.

8.3.4 Měření provedená na operačním systému FreeRTOS

Na operačním systému FreeRTOS bylo provedeno několik časových měření za účelem zjištění vytížení mikrokontroléru při zpracování systémových operací, jako jsou:

- doba přepnutí kontextu
- doba zamknutí a odemknutí mutexu
- zápis do fronty a čtení z fronty zpráv
- doba pro vytvoření vlákna

Mikrokontrolér MC9S12DP512, s vnitřní frekvencí sběrnice 25MHz, byl testován s verzí FreeRTOS 6.0.2. V tabulce níže jsou uvedeny výsledné doby trvání a jejich procentuální vyjádření vztažené k délce trvání systémového ticku o délce 1 ms. Tím je zjištěna velikost režie operačního systému vyjádřená v procentech.

Tab. 7 Časové měření na FreeRTOS

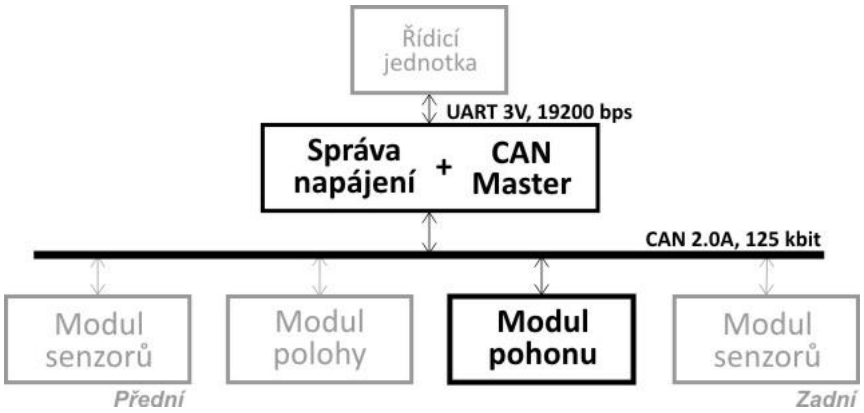
Operace	Doba provedení [µs]	% času procesoru
Přepnutí kontextu	8,25 ÷ 16	0,82 ÷ 1,6
Vytvoření vlákna	158,8	15,9
Zamknutí mutexu	22,55 ÷ 6,15	2,25 ÷ 0,62
Odemknutí mutexu	6,95 ÷ 15,8	0,7 ÷ 1,6
Zápis do fronty zpráv	22,2 ÷ 40,2	2,2 ÷ 4
Čtení z fronty zpráv	15,4 ÷ 19,4	1,5 ÷ 2

Z naměřených výsledků je zřejmé, že nedochází k rapidnímu nárůstu režie při použití operačního systému FreeRTOS. Časové nároky systémových operací jsou minimální.

Nutno podotknout, že výběr operačního systému a jeho úprava pro mikrokontrolér Freescale HCS12 byl společnou prací celkově dvou diplomantů. Jeho samotná implementace do cílových zařízení (modulů) již byla předmětem samostatné práce.

9 Komunikace

Průzkumné vozidlo využívá dvě různá komunikační rozhraní. Každé z nich má odlišné nároky na cyklus sběrnice, časování, míru spolehlivosti přenosu, objem přenesených dat apod. Schéma komunikací mezi moduly lze vidět na následujícím obrázku Obr. 33.



Obr. 33 Komunikace na jednotlivých úrovních

Na vrcholu hierarchického distribuovaného řídicího systému se nachází modul řídicí jednotky i.Mx31. Ten komunikuje s podřízenou jednotkou správy napájení pomocí klasické 3V sériové linky UART na komunikační rychlosti 19200 Bd. Modul správy napájení dále komunikuje s podřízeným systémem po sběrnici CAN na rychlosti 125 kbit. Modul tedy tvoří datovou bránu mezi podřízeným systémem se sběrnicí CAN a nadřazeným systémem s komunikací UART. Hlavní rysy komunikací:

a) podřízený systém se sběrnicí CAN
- rychlá výměna a sběr procesních dat
- malý objem přenášených dat
- deterministický a spolehlivý přenos dat, v případě rušení se data opětovně vyšlou

b) nadřazený systém s komunikací UART
- menší nároky na přenosovou rychlost a deterministické chování
- větší objem přenášených dat
- cyklická výměna dat, v případě rušení se neplatná data zahazují

Návrh komunikací a komunikačních protokolů byl společnou prací celkově dvou diplomantů, avšak její samotná implementace do cílových zařízení (modulů) již byla předmětem samostatné práce.

9.1 Rozhraní UART

UART je sériová asynchronní komunikace s nejčastěji 8bitovou délkou užitečné informace, dále jedním start bitem a jedním stop bitem. Fyzická vrstva je tvořena metalickým vedením obsahujícím vodič pro vysílání (Tx), vodič pro příjem (Rx) a zem (GND). Nejčastěji používané komunikační rychlosti se pohybují v rozmezí od 9600 Bd do 115200 Bd. Komunikace je vlivem nesymetrického vedení a absencí mechanismů pro detekci chyby náchylná k rušení, proto se často používají komunikační rychlosti do 19200 Bd.

9.1.1 Implementace komunikace prostřednictvím rozhraní UART

Pro komunikaci prostřednictvím rozhraní UART mikrokontroléru Freescale HCS12 s použitým operačním systémem reálného času FreeRTOS bylo potřeba napsat knihovny pro inicializaci, odesílání a příjem zpráv. To vše za pomocí systémových nástrojů (vlákna, synchronizační nástroje), které poskytuje operační systém, a konfiguračních registrů a přerušení, které poskytuje mikrokontrolér.

Inicializace rozhraní UART

Pro inicializaci registrů komunikace slouží funkce:

```
void vSCI0_init(UINT32 speed);
```

Jejími vstupními parametry jsou:

- *speed* – rychlost komunikace v Baudech

Při inicializaci se provede nastavení rychlosti komunikace, povolení přerušení pro dokončení příjmu a odeslání. Dále se zde vytvoří fronty zpráv pro příjem a odesílání zpráv. Tyto fronty pak budou tvořit vyrovnávací buffer pro zprávy, které čekají na zpracování nebo odeslání operačním systémem.

Odeslání a příjem dat po rozhraní UART

Pro odeslání dat se využívá zápisu do fronty zpráv. Fronta zpráv ale neobsahuje odesílaná data, ale pouze ukazatele na místo v paměti, kde se odesílaná data nacházejí. Data jsou tedy fyzicky uložena ve dvourozměrném poli *UartTxBuffer[][]*, kde jednotlivé řádky pole jsou odesílané zprávy čekající na odeslání.

Metoda fronty ukazatelů je velmi výkonná oproti časově náročnému kopírování dat do fronty.

Příjem dat v přerušení pracuje na stejném principu, data jsou pak fyzicky uložena v dvourozměrném poli *UartRxBuffer[][]*.

Do sestavování odesílané zprávy a kontrolování správnosti přijaté zprávy však zasahuje vyšší vrstva, která bude popsána v následující podkapitole.

9.1.2 Vyšší vrstva pro rozhraní UART

Pro komunikaci po sériové lince je potřeba dát datům určitou formu, určitý řád. Proto byl sestaven jednoduchý komunikační protokol, který odesílá data ve „zprávách“. Zprávy mívají různou délku a různý obsah, proto je potřeba, aby nesly informaci o délce a typu dat. To umožňuje identifikaci přijaté zprávy a její následné dekódování.

Každá zpráva na sériové lince má následující formát:

<STX> <LBH> <LBL> <TYP> <…….DATA…….> <ETX>

kde

<STX> - (Start of transmit) začátek rámce, znak 0x02 v ASCII

<LBH> - (Length Byte High) horní Byte délky zprávy

<LBL> - (Length Byte Low) dolní Byte délky zprávy

<TYP> - typ zprávy, identifikátor přenášených dat

<DATA> - n Bytů přenášených dat

<ETX> - (End of transmit) konec rámce, znak 0x03 v ASCII

Bylo vytvořeno celkem 58 zpráv, každá se svým unikátním identifikátorem, kde nejdelší zpráva má délku okolo 250 Bytů. Je zde však ponechána rezerva v případě potřeby delších zpráv (Byte LBH). Seznam zpráv je uveden v přílohách *Příloha II - Tabulka zpráv pro komunikaci s nadřazeným systémem i.MX31*.

Každá zpráva na sběrnici obsahuje užitečná data jedné proměnné. To znamená, že je vytvořena zvlášť zpráva pro 16bitovou hodnotu *PredniUltrazvuk*, zvlášť pak pro 8bitový údaj *NatoceniKol* atd. To sice zvyšuje režii při sestavování a dekódování zprávy, ale v případě krátkého výpadku komunikace bude ztracena jen aktuální zpráva na rozdíl od přenosu všech proměnných v jedné velké zprávě.

Zprávy se vysílají ve dvou různých intervalech. Ty důležité jsou odesílány se 100ms intervalem, ty méně důležité pak s intervalem 1 sekunda.

Odesílání zpráv na rozhraní UART

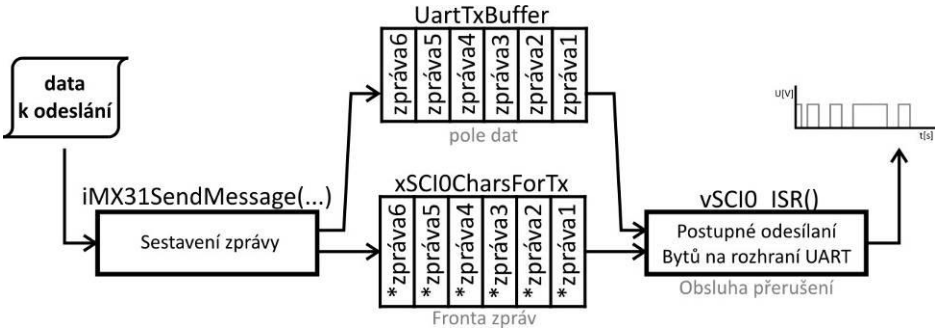
Pro odesílání zpráv po sériové lince se využívá funkce:

*void iMX31SendMessage(UINT8 type,void * data, size_t length, UINT8 datatype)*

Jejími vstupními parametry jsou:

- *type* – identifikátor zprávy, umožňuje vysílači dekódovat zprávu, číslo v rozsahu 0 až 58
- **data* – ukazatel na data k odeslání
- *length* – délka odesílaných dat
- *datatype* – datový typ odesílaných dat (char, int, long, pole char, pole int, …)

Pro objasnění principu odesílání je uveden spolu s popisem funkce i Obrázek Obr. 34. Po vstupu do funkce *iMX31SendMessage* se začne skládat tvar zprávy pro odeslání. Na příslušný volný řádek dvourozměrného pole dat *UartTxBuffer,* tvořícího buffer zpráv pro odeslání, se začnou zapisovat znaky STX, LBH, LBL atd. Po dokončení zápisu dat a ukončovacího znaku STX se ukazatel na zprávu odešle do fronty zpráv pro odeslání a povolí se přerušení od dokončení vyslání na UART kanále. Prakticky okamžitě se začnou (na pozadí procesů) v přerušení posílat data na sériovou linku. To trvá do doby, než jsou všechny Byty zprávy odeslány. Poté se povolení přerušení pro odeslání opět zakáže.

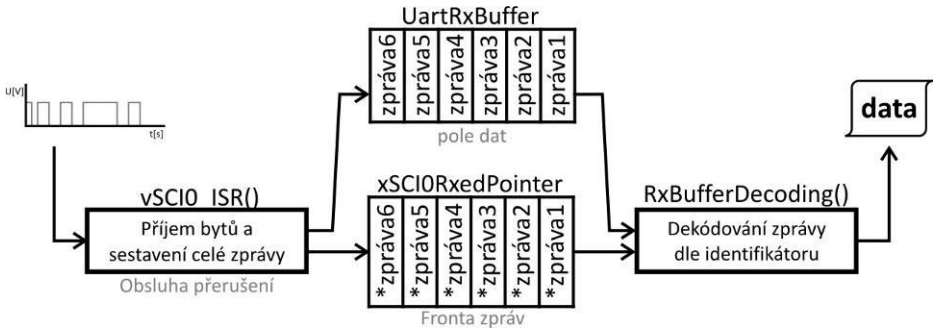


Obr. 34 Schéma odesílání dat UART komunikace

Nutno upozornit na jednu skutečnost, která vyvstala během řešení komunikace procesoru HCS12 s nadřazeným systémem s procesorem i.Mx31. Procesory používají navzájem jinou endianitu. Jde o způsob uložení čísel v paměti procesoru. Zatímco HCS12 je typu Big-endian, procesor i.Mx31 používá Little-endian. To přináší do komunikace další ztížení, kdy se pořadí Bytů datových typů delších než 1 Byte musí otočit. Před zápisem užitečných dat do bufferu *UartTxBuffer* je tedy potřeba provést konverzi pořadí Bytů. K tomu slouží informace *datatype,* která určuje, jak bude konverze probíhat.

Přijímání zpráv na rozhraní UART

Znakově orientovaný přenos dat, jakým je sériová linka, vyžaduje pro příjem zpráv paměť (buffer), do kterého se budou přijaté znaky zapisovat až do přijetí posledního znaku zprávy, jímž je znak ETX. Bufferem je v tomto případě dvourozměrné pole *UartRxBuffer*, do kterého se v přerušení přijímají znaky přijímané zprávy a který obsahuje i všechny ostatní přijaté, ale nezpracované zprávy.



Obr. 35 Schéma příjmu dat UART komunikace

Během příjmu dat se kontroluje jejich správnost, délka a v případě jakékoliv chyby se data zahodí a čeká se na nový platný paket začínající znakem STX. Ztráta dat v případě jejich zahození dat není závažná, neboť perioda komunikace je 100 ms a nová platná data se tedy přijmou s minimálním zpožděním.

Po dokončení příjmu posledního znaku zprávy (ETX) se zkopíruje ukazatel na místo uložení zprávy v poli *UartRxBuffer* do fronty zpráv. Zpráva je tedy kompletní a je připravena k dekódování.

Dekódování se provádí ve vláknu, které přečte z přijímací fronty ukazatel na přijatou zprávu a podle jejího identifikátoru následně provede zkopírování dat ze zprávy do příslušné proměnné využívané v programu Modulu správy napájení.

Rovněž zde musel být řešen problém s rozdílnou endianitou komunikujících architektur, viz *Odesílání zpráv na rozhraní UART*.

9.2 Sběrnice CAN

CAN je sériová komunikační sběrnice vyvinutá firmou Bosch pro automobilový průmysl, která se vyznačuje relativně vysokou přenosovou rychlostí, odolností proti rušení, nadprůměrnou spolehlivostí, a také nízkou cenou komunikačních obvodů. Převážně dobré vlastnosti zapříčinily její rozšíření i do mnoha dalších oborů, než pro které byla původně vyvinuta.

Z důvodů velké spolehlivosti přenosu byla zvolena pro komunikaci s podřazeným distribuovaným systémem, který se vyznačuje malým objemem přenášených dat, ale za to rychlou požadovanou odezvou sběrnice.

Sběrnice CAN využívá metodu přístupu k médiu CSMA/BA (někdy také značeno CSMA/CR), tedy CSMA s bitovou arbitráží. Každá zpráva vyslaná na CAN, jejíž formát je vidět na obrázku Obr. 36, obsahuje 11bitový identifikátor zprávy a právě tyto bity se podílejí na bitové arbitráži. Nejvyšší prioritu má pak identifikátor 0x000, nejnižší pak 0x800.

Obr. 36 Formát zprávy s 11bitovým identifikátorem

Každý rámec začíná start bitem s hodnotou v logické nule. Za ním následuje identifikátor, který slouží k adresování, a spolu s bitem RTR určuje také prioritu vysílané zprávy. Následující bit IDE (Identifier extension), určuje, zda se jedná o zprávu se standardním 11bitovým, nebo rozšířeným 29bitovým identifikátorem, dle standardu 2.0A či 2.0B. Za bitem r0, který je rezervován pro budoucí rozšíření, následují 4 bity určující délku datového pole v bajtech, což je maximálně 8 bajtů. Za datovým polem je pole CRC (Cyclic Redundancy Code), jež se používá jako bezpečnostní kontrola na detekci bitových chyb v celé předcházející části zprávy. Pole ACK (Acknowledge field) obsahuje dva bity. Vysílač vyšle první bit jako „vrátný“ s logickou úrovní 1, a pokud alespoň jeden z přijímačů přijal zprávu v pořádku, přepíše tuto úroveň na logickou 0. Druhý

bit je vždy vysílán v logické jedničce. Konec zprávy EOF (End of frame) obsahuje 7 bitů v logické jedničce a je následován třemi bity IFS (Inter Frame Space), mezilehlým polem pro uklidnění stavu na sběrnici.

9.2.1 Implementace komunikace prostřednictvím sběrnice CAN

Pro komunikaci prostřednictvím sběrnice CAN mezi mikrokontroléry Freescale HCS12 s použitým operačním systémem reálného času FreeRTOS bylo potřeba napsat knihovny pro inicializaci, odesílání a příjem zpráv. To vše za pomocí systémových nástrojů (vlákna, synchronizační nástroje), které poskytuje operační systém, a konfiguračních registrů a přerušení, které poskytuje mikrokontrolér.

Inicializace sběrnice CAN

Pro inicializaci registrů komunikace slouží funkce:

```
void CAN0Init(UINT16 rychlost, UINT8 sizeofQueue, CAN_FILTR *FiltrMaska);
```

Jejími vstupními parametry jsou:

- *rychlost* – rychlost komunikace, která se může pohybovat v rozsahu 20kbit až 1Mbit
- *sizeofQueue* – velikost fronty zpráv jak pro vysílání, tak pro příjem
- **FiltrMaska* – ukazatel na strukturu obsahující nastavení masek a filtrů pro řadič CAN

Při inicializaci se provede nastavení rychlosti komunikace, filtrů a masek. Dále se vytvoří fronta zpráv pro příjem a vysílání zpráv. Tyto fronty slouží jako buffer pro uchování zpráv, které čekají na zpracování nebo odeslání operačním systémem.

Odeslání na sběrnici CAN

Pro odesílání dat se používají funkce:

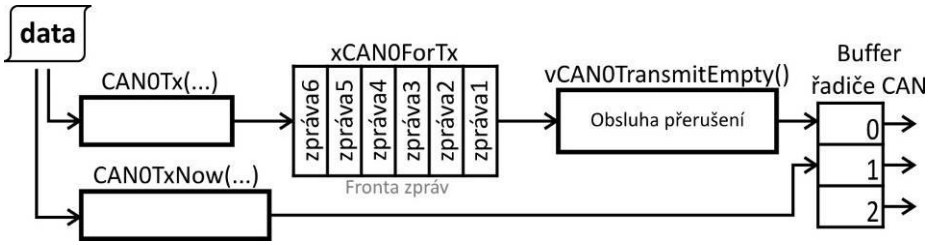
```
void CAN0Tx( CANTX *odeslat );

void CAN0TxNow( CANTX *odeslat );
```

Jejími vstupními parametry jsou:

- **odeslat* – ukazatel na strukturu dat připravených k odeslání

Pro standardní posílání CAN zpráv slouží funkce *CAN0Tx(...)*. Ta provede zápis dat z datové struktury do fronty zpráv pro odeslání a nastaví povolení přerušení od dokončení vysílání na sběrnici CAN. Jakmile je vysílací registr prázdný (minulá zpráva odeslána), vyvolá se přerušení *vCAN0TransmitEmpty()*, ve kterém se data z fronty zkopírují do vysílacích registrů řadiče CAN. Na konci přerušovací rutiny se přerušení opět zakáže.

Na obrázku Obr. 37 lze vidět, že samotný CAN řadič obsahuje tříúrovňový prioritní buffer, kde nejmenší prioritu má pozice 0 a největší pozice 2. Standardní funkce *CAN0Tx(…)* používá výhradně pozici 0.



Obr. 37 Schéma odesílání dat na sběrnici CAN

Pro urgentní časově kritické data by posílání frontou znamenalo velké opoždění ve vyslání. Proto je implementována funkce *CAN0TxNow()*, která umožní zápis na 1. pozici bufferu CAN řadiče, a tím dojde k okamžitému vyslání dat.

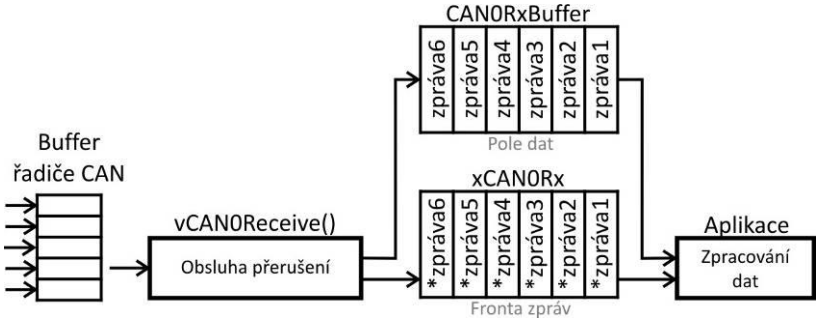
Fronta zpráv má pro odesílání výbornou vlastnost. Po jejím zaplnění neumožní další zápis dat a operační systém nepřidělí vláknu s odesláním nových dat do fronty výpočetní čas, dokud se ve frontě opět neuvolní místo.

Příjem na sběrnici CAN

Data na sběrnici CAN se po úspěšném průchodu maskami a filtry přijímají v pětiúrovňovém přijímacím bufferu řadiče CAN. Pokud je tento naplněn, další příjem zprávy se vysílači nepotvrdí a to má za následek opakovaného vysílání zprávy vysílačem, dokud některý z uzlů na CANu zprávu nepotvrdí.

Při příjmu dat z CANu a uložení do bufferu řadiče se vyvolá přerušení od příjmu *vCAN0Receive()*. V přerušovací rutině se přijatá data zkopírují do datového úložiště a do přijímací fronty zpráv se zkopíruje pouze ukazatel na místo v datovém úložišti, kde jsou data uložena. To rapidně snižuje časové nároky na operaci s daty při jejich zpracování, ať už v přerušení, tak v následném vláknu pro zpracování.

Pokud dojde k naplnění přijímací fronty zpráv, zakáže se přerušení a přijatá zpráva se ponechá v registrech CAN řadiče. Následující nepřijatá data se „neztratí“, neboť se nevyšle vysílači potvrzení o příjmu. Toto blokování trvá až do chvíle, kdy se v přijímací frontě zpráv neuvolní místo.



Obr. 38 Schéma příjmu dat na sběrnici CAN

9.3 Komunikační protokol CANopen

CANopen je vyšší komunikační protokol využívající ve většině případů sběrnici CAN. Definuje komunikační profily a rozhraní pro komunikaci s uživatelskou aplikací. Protokol CANopen je velmi flexibilní a poskytuje mnoho možností nastavení a využití. Komunikační protokol obsahuje podporu pro správu sítě, monitoring a komunikaci mezi uzly, včetně jednoduché transportní vrstvy pro správu segmentace / desegmentace. Protokol CANopen umožňuje vývojáři vyhnout se řešení problémů specifických pro CAN, jako je např. správné časování zpráv. Dosahuje se toho zavedením standardních komunikačních objektů: PDO (Proces Data Objects) pro časově kritická (real-time) data, SDO (Service Data Objects) pro konfigurační zprávy a dalších pro speciální funkce (časování, synchronizaci a mimořádné situace) a pro síťové služby (nové spuštění zařízení, správu sítě a chybové zprávy). [4]

Kompletní popis protokolu CANopen je k dispozici v dokumentaci s názvem *CiA Draft Standard 301* na internetových stránkách [3] , která sdružuje výrobce a uživatele tohoto komunikačního standardu.

9.3.1 Komunikační objekty

Základním prvkem CANopen je pojem komunikační objekt. Komunikační objekt je základní jednotka přenášející data mezi zařízeními komunikujícími podle protokolu CANopen. Podle svého určení se objekty dělí do skupin, z nichž nejdůležitější jsou:

- technologické objekty, tzv. PDO (Process Data Objects), které nesou informace o technologických veličinách (teplota, otáčky, napětí apod.), jejich délka je až osm bajtů a mají v síti velkou prioritu,
- servisní objekty, tzv. SDO (Service Data Objects), které nesou servisní nebo konfigurační údaje (počet I/O, konstanty PID regulátoru, měřicí rozsah snímače apod.), jejich délka může být libovolně velká a mají v síti malou prioritu,
- objekty pro správu sítě, tzv. objekty NMT (Network Management), které slouží ke konfigurování a řízení provozu sítě s protokolem CANopen (hlášení stavu zařízení, povel k novému spuštění zařízení apod.) a mají v síti největší prioritu,
- objekty pro synchronizaci zpráv na síti, tzv. SYNC (Synchronization), které slouží pro vyzvání k odeslání dat z ostatních uzlů na síť,
- objekty pro zjištění stavů komunikačních uzlů na síti, tzv. Heartbeat a Node Guarding, pomocí kterých jsme schopni zjistit, zdali daný modul na síti je aktivní a v pořádku či nikoliv.

Jako standardní typ zprávy se využívá CAN zpráva dle specifikace 2.0A, teda zpráva s 11bitovým identifikátorem. Celý identifikátor se v CANopen také označuje jako COB-ID (Communication Object Identifier).

Identifikátor je rozdělen na dvě pole. První tvoří čtyři horní bity (10 až 7), které se označují jako kódy funkcí (Function Code) a zbylé bity pak tvoří druhou část označovanou jako „Node-ID“, neboli adresa zařízení.

Následující tabulka (Tab. 8) shrnuje zprávy, které se na sběrnici mohou vyskytnout. První sloupec ukazuje pro jaký objekt je daný rozsah používán, druhý pak ukazuje, jakou hodnotu mají jejich kódy funkcí, a poslední sloupec zobrazuje rozsah identifikátoru.

Tab. 8 Zprávy protokolu CANopen (podle identifikátoru) [3]

Objekt	Kód funkce (binárně)	COB-ID
NMT	0000	0
SYNC	0001	128(80h)
TIMESTAMP	0010	256(100h)
EMERGENCY	0001	129(81h) – 255(FFh)
PDO1 (tx)	0011	385(181h) – 511(1FFh)
PDO1 (rx)	0100	513(201h) – 639(27Fh)
PDO2 (tx)	0101	541(281h) – 767(2FFh)
PDO2 (rx)	0110	769(301h) – 895(37Fh)
PDO3 (tx)	0111	897(381h) – 1023(3FFh)
PDO3 (rx)	1000	1025(401h) – 1151(47Fh)
PDO4 (tx)	1001	1153(481h) – 1279(4FFh)
PDO4 (rx)	1010	1281(501h) – 1407(57Fh)
SDO (tx)	1011	1409(581h) – 1535(5FFh)
SDO (rx)	1100	1537(601h) – 1663(67Fh)
NMT Error Control	1110	1793(701h) – 1919(77Fh)

9.3.2 Slovník objektů

Komunikační objekty jsou zařazeny v tzv. slovníku objektů (Object Dictionary) uloženém v zařízení, které je součástí sítě, a sloužícím jako rozhraní mezi samotným zařízením a aplikačním programem. Všechny komunikační objekty příslušné určitému zařízení (údaje specifické pro aplikaci a konfigurační parametry) jsou ve slovníku objektů popsány předepsaným způsobem v pevně stanoveném formátu. Každý komunikační objekt je dostupný prostřednictvím šestnáctibitového indexu, v případě objektů typu polí a záznamů (objektů složených s několika dalších objektů) doplněného navíc osmibitovým subindexem. Slovník objektů tvoří prostředníka mezi uživatelskou aplikací a samotnou CANopen komunikací. [3]

9.3.3 Implementace CANopen protokolu

Pro komunikaci modulů na vozidle, nebylo třeba implementovat všechny části specifikace CANopen. Byly vytvořeny funkce a vlákna, které implementují tyto protokoly: PDO, SYNC a Heartbeat. Ve slovníku objektů byly vytvořeny pouze nezbytné záznamy, které jsou popsány v následující tabulce Tab. 9. Tyto záznamy musí podporovat každé CANopen zařízení na síti.

Aby mohl modul komunikovat s okolím, musí být nadefinovány PDO zprávy (viz dále), které obsahují procesní proměnné jak vysílané, tak přijímané modulem. Tyto jsou uvedeny v souhrnném přehledu v příloze *Příloha I - Slovník komunikačních objektů CANopen*.

Tab. 9 Nezbytné záznamy ve slovníku objektů pro CANopen komunikaci [3]

Index	Název	Popis
1000h	DeviceType	Typ zařízení a jeho funkce
1001h	ErrorRegister	Místo pro ukládání vnitřních chyb zařízení
1005h	COBIDSyncMessage	Nastavení protokolu SYNC
1006h	CommCyclePeriod	Nastavuje periodu generování zprávy SYNC
1016h	ConsumHeartbeatTime	Nastavuje zařízení na Heartbeat konzumenta, přičemž definuje periodu vysílání pro jednotlivá zařízení na sběrnici
1017h	ProducerHeartbeatTime	Nastavuje periodu vysílání zprávy Heartbeat protokolu
1018h	IdentityObject	Obsahuje obecné informace o zařízení
1400 – 1407h	RPDOCommPar	Nastavuje jaké PDO zprávy se budou přijímat
1600 – 1607h	RPDOMappingPar	Nastavuje, jaké proměnné se v přijatých zprávách nacházejí
1800 – 1807h	TPDOCommPar	Nastavuje jaké PDO zprávy se budou vysílat
1A00 – 1A07h	TPDOMappingPar	Nastavuje, jaké proměnné jsou ve vysílaných zprávách

Inicializace CANopen protokolu

Pro inicializaci se využívá funkce:

```
void vCANopenInit( void );
```

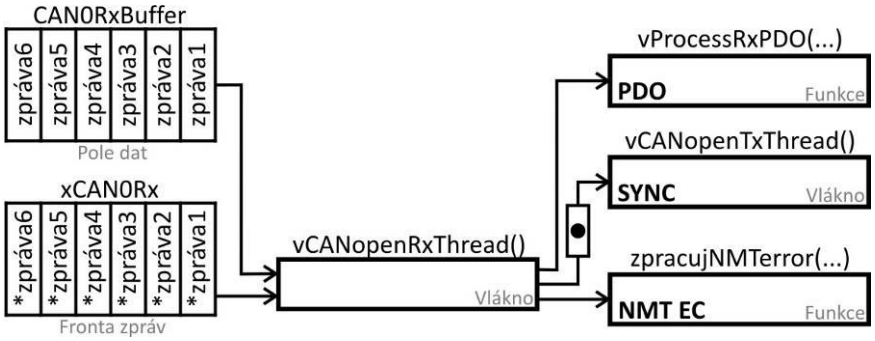
Ta by měla být provedena ještě před spuštěním plánovače operačního systému. Při inicializaci se provádí následující kroky:

- Inicializuje se slovník objektů, tzn., nastaví se jeho výchozí parametry
- Vytvoří se synchronizační nástroje potřebné pro jednotlivé protokoly
- Provede se inicializace řadiče sběrnice CAN
- Zjistí se, kolik RPDO komunikačních parametrů je nadefinováno. Ty se pak budou kontrolovat při příjmu PDO zprávy
- Zjistí se, kolik TPDO komunikačních parametrů je nadefinováno. Ty se pak budou vysílat
- Vytvoří se vlákno, které zajišťuje přebírání přijatých zpráv ze sběrnice CAN a jejich další třídění (*vCANopenRxThread*)
- Pokud je nadefinována alespoň jedna zpráva k vysílání, je vytvořeno vlákno obstarávající synchronní vysílání zpráv (*vCANopenTxThread*)
- Pokud je v objektu 1005h nadefinováno generování zprávy SYNC, vytvoří se pro její generování vlákno (*vCANopenTxSYNCThread*)
- Pokud má objekt 1017h nenulovou hodnotu (je nadefinována perioda vysílání) vytvoří se vlákno (*vCANopenHeartBeatProducer*) pro vysílání zpráv protokolu Heartbeat, v opačném případě se zařízení stává Heartbeat konzument a vytváří se patřičné vlákno (*vCANopenHeartBeatConsumer*)

Maximálně se tedy pro CANopen vytváří čtyři vlákna.

Příjem a třídění zpráv

Pro příjem a roztřídění příchozích zpráv je využito vlákno s názvem *vCANopenRxThread*. Třídění se provádí pomocí hodnoty identifikátoru zprávy, hlavně částí kódů funkcí. Na následujícím obrázku Obr. 39 je znázorněno, jak zprávy procházejí tříděním až ke svému zpracování.



Obr. 39 Třídění zpráv protokolem CANopen

Vlákno pro třídění čte data z fronty zpráv *xCAN0Rx*. Přijetí zprávy SYNC pro systém znamená pokyn k odeslání PDO zpráv s nastaveným typem vysílání na synchronní. To se zajišťuje inkrementací binárního semaforu *xSemTxSYNC*, který odesílání spustí. V případě přijetí zprávy typu PDO se vyvolá funkce *vProcessRxPDO()*, ve které se podle RPDO komunikačních a mapovacích parametrů (ve výchozím stavu objekty 1400h až 1407h a 1600h až 1600h) uloží přijaté procesní data. Pokud je zařízení nastaveno jako konzument Heartbeat protokolu přijímá i zprávy typu NMT Error Control. Tyto zprávy zpracovává funkce *zpracujNMTError()*, ve které se ukládají časy přijetí zpráv od jednotlivých Heartbeat producentů.

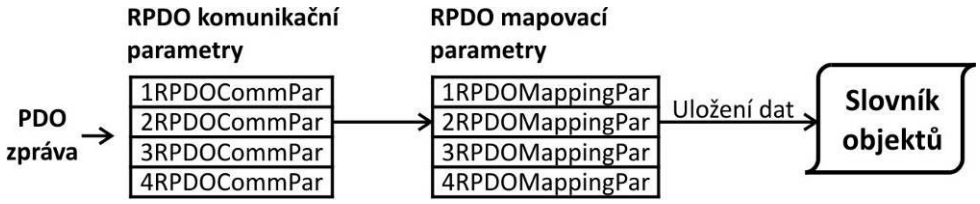
SYNC

K nastavení vysílání zprávy SYNC se využívají objekty 1005 a 1006h. V prvním objektu se vysílání aktivuje/deaktivuje a v druhém se nastavuje perioda vysílání v mikrosekundách. V našem případě je nastavení periody omezeno na násobky milisekund s nejmenší nenulovou hodnotou jedné milisekundy (1000uS). Periodické vysílání zajišťuje vlákno *vCANopenTxSYNCThread*, které využívá funkce *CAN0TxNow(...)*. Ta nepoužívá vysílací frontu zpráv, ve které mohou být i zprávy s nízkou prioritou, ale přistupuje přímo k řadiči MSCAN, čímž je zachována co největší přesnost periody vysílání.

Při příjmu zprávy SYNC se pouze inkrementuje semafor, na který čeká vlákno pro periodické (synchronní) vysílání PDO zpráv.

PDO

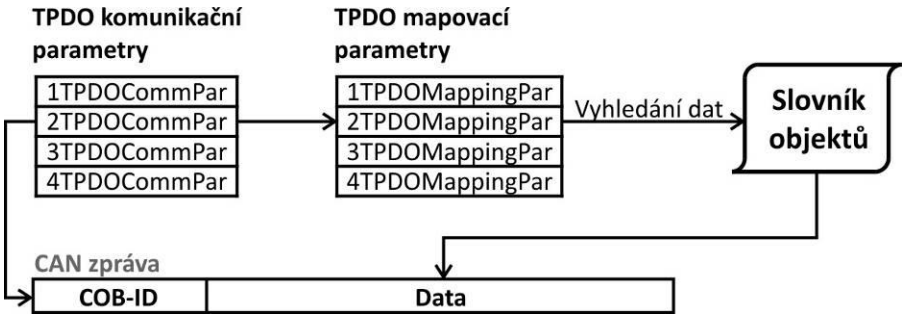
Po přijetí zprávy typu PDO se vyvolá funkce *void vProcessRxPDO()*, ve které se rozhodne, jestli přijatá zpráva je určena právě pro toto zařízení. To jak funkce rozhoduje, zda se má zpráva přijmout, je nastíněno na následujícím obrázku Obr. 40. Shoda identifikátoru zprávy s RPDO komunikačním parametrem znamená, že zpráva je určena pro toto zařízení. Následně se vyhodnotí RPDO mapovací parametr, který nese informace o počtu a rozložení jednotlivých proměnných ve zprávě, které se následně uloží na své pozice ve slovníku objektů.



Obr. 40 Postup při dekódování přijaté PDO zprávy

CANopen specifikuje několik typů událostí pro vysílání PDO zpráv. Pro potřeby vozidla bylo nutné implementovat vysílání na událost příjmu zprávy SYNC (synchronní) a také na události, které vznikají uvnitř systému.

Zařízení může kdykoli PDO zprávu vyslat voláním funkce *void vSendTxPDO(...)*. Tato funkce má vstupní argument index (ze slovníku objektů) odpovídající TPDO komunikačnímu parametru zprávy, která se má vyslat na sběrnici. Zpráva se složí z TPDO komunikačního parametru a proměnných v pořadí, které je určené TPDO mapovacím parametrem. Sestavení zprávy je ukázáno na následujícím obrázku Obr. 41.



Obr. 41 Postup při odesílání PDO Zpráv

Synchronní vysílání zpráv zajišťuje vlákno *vCANopenTxThread*, které při příjmu zprávy SYNC prochází všechny TPDO komunikační parametry, ve kterých je určeno, zda a jak často (kolik SYNC zpráv musí přijít) se příslušná zpráva vysílá.

Heartbeat Protokol

Pokud je zařízení nastaveno jako producent Heartbeat zpráv (objekty 1016h a 1017h), je spuštěno vlákno *vCANopenHeartBeatProducer*, ve kterém se s nastavenou periodou vysílají na sběrnici zprávy informující mastera (zařízení pracující jako Heartbeat konzument) o aktuálním stavu zařízení.

V případě, že je zařízení nastaveno na konzumenta Heartbeat zpráv je spuštěno vlákno *vCANopenHeartBeatConsumer*, ve kterém se periodicky kontrolují odezvy jednotlivých zařízení na sběrnici. Pokud nastane situace, kdy od jednoho zařízení nepřijdou dvě po sobě jdoucí zprávy v nastaveném časovém intervalu, vyhodnotí se to jako chyba vysílání tohoto zařízení, a voláním funkce *void ProcessHeartBeatError(...)* se dá aplikaci vědět, které zařízení má problémy.

10 Softwarový návrh

Pro programování je použito BDM rozhraní, které komunikuje s programátorem P&E Multilink. Samotný P&E Multilink pak komunikuje s PC pomocí sběrnice USB.

Pro editaci programu byl použit program Freescale CodeWarrior IDE v5.90.

10.1 Modul pohonu

Program v modulu pohonu je rozdělen na jednotlivá vlákna dle jejich funkce a jejich periodě volání. Vlákna můžou nabývat hodnoty priority od nejmenší (0) po největší (4). V následujícím přehledu jsou vlákna seřazena dle jejich priority od největší po nejmenší:

- **vCpuTimeThread –** vlákno pro sledování a výpočet vytížení procesoru, voláno s periodou 1 sekunda.
- **vPIDLoopControl –** voláno cyklicky s periodou 10 ms a provádí operace spojené s řízením pohonu.
- **vAnalogMeasure –** voláno cyklicky s periodou 5 ms, provádí měření, kalibraci a filtraci analogových hodnot potřebných pro řízení pohonu.
- **vLedTask –** vlákno informující o stavu operačního systému FreeRTOS, voláno cyklicky s periodou 500 ms.

Kromě těchto vláken zde běží vlákna pro CANopen komunikaci, tyto jsou ale uživateli skrytá a proto nejsou zahrnuta ani v přehledu. Jejich popis nalezneme v kapitole 9. Komunikace.

Modul pohonu je CANopen heartbeat producer (slave), tudíž produkuje zprávy s periodou 500 ms, které slouží CANopen heartbeat consumer (master) pro informaci o správnosti chodu modulu.

Slovník objektů využívaných modulem pohonu je pro svou velikost uveden v přílohách. Všechna data ze slovníku objektů jsou následně uložena ve struktuře *DriveControl*. Tato struktura obsahuje:

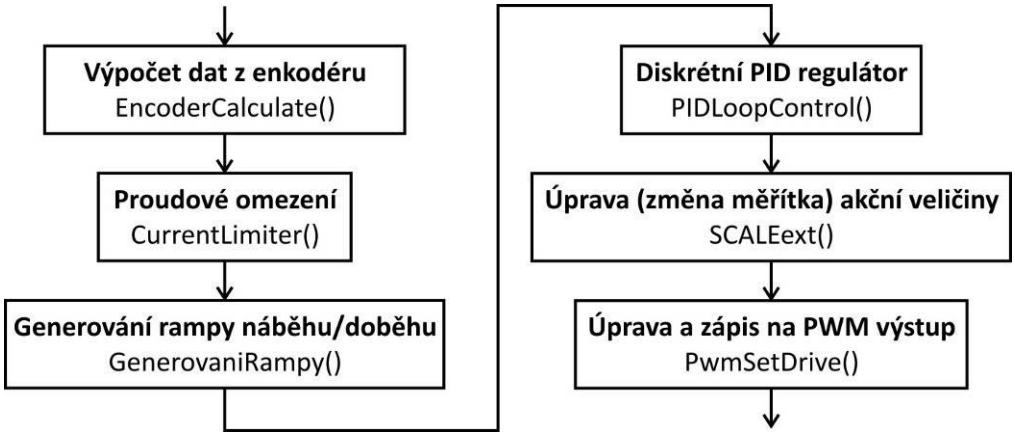
- data enkodéru, jeho nastavení a aktuální hodnota snímaných otáček a směru pohybu
- data motoru, jeho aktuální a požadované otáčky, proud tekoucí motorem
- data pro aktivaci brzdy, nastavení horní a dolní meze servomotoru pro brzdu
- data pro generování rampy pro náběh a doběh motoru

Protože se ke struktuře *DriveControl* přistupuje z více vláken, je potřeba k nim řešit výlučný přístup pomocí nástroje mutex. Jakýkoliv zápis nebo čtení dat struktury je chráněn mutexem *xMutexGlobalData*.

V následujících podkapitolách budou podrobněji popsána jednotlivá vlákna modulu správy energie.

10.1.1 Vlákno vPIDLoopControl

Je nejdůležitějším vláknem modulu pohonu. Obsahuje volání funkce *DriveControlTask()* s periodou 10 ms, ve které se provádí algoritmus pro řízení pohonu, uvedený na obrázku Obr. 42.



Obr. 42 Vlákno vPIDLoopControl

- **EncoderCalculate()** – Časové značky snímané v přerušení od záchytného systému, se v této funkci přepočítávají na otáčky za minutu. Rovněž se zde zjišťuje směr otáčení motoru.

- **CurrentLimiter() –** Funkce hlídající velikost proudu tekoucího motorem. Do funkce vstupuje proměnná *požadované otáčky*. Pokud je překročena maximální nastavená mez, dojde k vynulování požadovaných otáček motoru. To má za následek poklesu proudu až k nule.

- **GenerovaniRampy() –** Funkce generující náběhovou a doběhovou rampu pro vstupní hodnotu *požadované otáčky*. Působí jako ochrana proti rychlé změně požadovaných otáček, což by znamenalo velké špičkové odběry pohonnou jednotkou. Jsou zde definovány zvlášť konstanty pro náběh a doběh. To umožňuje vytvořit rychlejší zastavení oproti pomalejšímu rozjezdu.

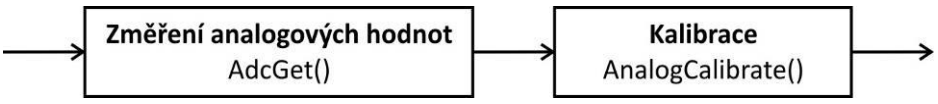
- **PIDLoopControl() –** Funkce obsahující diskrétní PID regulační algoritmus, včetně dynamického řešení anti-windupu a limitace akční veličiny na výstupu. Výstupem funkce je akční veličina v rozsahu $-7000 \div 7000$ otáček za minutu.

- **SCALEext() –** Protože vstupní údaj *akční veličina* má jiný rozsah, než který je možné zapisovat do registru střídy PWM, je zde použita tato funkce která převede hodnotu na rozsah $-100 \div 100$ %.

- **PwmSetDrive() –** Poslední neméně významná funkce vlákna obstarává zápis do registru střídy PWM. Protože může být napájecí napětí akumulátoru až 25 V a motor má jmenovité napětí 12 V, je potřeba aby maximální střída dosáhla hodnoty, která odpovídá efektivní hodnotě 12 V. Proto je napětí akumulátoru měřené správou energie a v závislosti na jeho aktuální hodnotě je upravována maximální hodnota střídy, aby k přetěžování motoru nedocházelo.

10.1.2 Vlákno vAnalogMeasure

Vlákno zprostředkovává cyklické měření analogových vstupů s periodou 5 ms. Po dokončení algoritmu na obrázku Obr. 43 se tedy vlákno „uspí“ a „probudí“ se za dalších 5 ms, kdy se celý algoritmus opakuje.



Obr. 43 Vlákno vAnalogMeasure

- **AdcGet()** – Funkce pro odměření analogových vstupů. Mezi měřené veličiny patří proud motoru a celkový proud tekoucí modulem pohonu. Tyto vstupy s napěťovým rozsahem $0 \div 5\,V$ jsou převedeny na 10bitovou hodnotu.

- **AnalogCalibrate()** – Naměřené analogové hodnoty je potřeba kalibrovat a převést je na jejich reálné fyzikální rozsahy. 10bitová hodnota pokrývá u použitých proudových čidel skutečný rozsah $-20 \div 20\,A$. Je tedy potřeba nalézt koeficienty K, Q a ty pak použít v kalibrační funkci. Kromě toho, vlivem různých nepřesností, bylo potřeba zkontrolovat skutečný proud odečtený z ampérmetru a porovnat ho s hodnotou naměřenou a případné korekce zanést rovněž do vztahu (22).

$$hodnota\_zkalibrovana = K \cdot hodnota + Q \tag{22}$$

10.1.3 Vlákno vLedTask

Vlákno pro vizuální kontrolu správnosti běhu operačního systému a potažmo celého modulu pohonu. Je zde využito dvou LED diod, jejichž střídavým blikáním s frekvencí 1 Hz je signalizován korektní stav modulu. Na následujícím výpisu programu vidíme kromě přepínání LED diod také funkci *vTaskDelay* poskytovanou operačním systémem, kterou se uspává vlákno na potřebnou dobu v milisekundách.

Tab. 10 Výpis programu – vlákno pro blikání LED diod

```
// ******************************************************* //
// ************************* LED task ********************** //
// ******************************************************* //
void vLedTask( void *pvParameters){

  for(;;){

    // zapis na vystup - LED diody
    CANopenRUN =~ CANopenRUN;
    CANopenERR =~ CANopenRUN;

    // uspani vlakna na dobu 500 ms
    vTaskDelay( 500 / portTICK_RATE_MS );
  }
}
```

10.1.4 Vlákno vCpuTimeThread

Toto vlákno slouží pro informaci o stavu vytížení mikrokontroléru a je udáno v procentech.

Mikrokontrolér nemusí v jistých okamžicích zpracovávat žádné vlákno. To znamená, že ani jedno z vláken není ve stavu *připraven* (Ready) a nezpracovává se žádné přerušení a procesor tedy nemá co na práci. Pro takové okamžiky slouží funkce *vApplicationIdleHook*, která se volá vždy ve „volných chvílích" mikrokontroléru. Tato funkce společně s vláknem *vCpuTimeThread* slouží pro výpočet vytížení procesoru. Na následujícím výpisu kódu lze vidět obě funkce.

Tab. 11 Výpis programu – vlákno a funkce pro výpočet % času procesoru

```
// ********************************************************* //
// *************** Vypocet % casu procesoru *************** //
// ********************************************************* //
void vCpuTimeThread(void){
  CpuTime = 0;

  for(;;){

    // uspani vlakna na dobu 1 s
    vTaskDelay( 1000 / portTICK_RATE_MS );

    // vypocet % casu procesoru
    CpuTime = (-0.003017046312 * IdleHookCitac + 100.0);
    IdleHookCitac = 0;
  }
}

// ********************************************************* //
// ****************** Idle hook task ********************** //
// ********************************************************* //
void vApplicationIdleHook(void){
    int i;

    // pro zmenseni poctu inkrementaci je zde zarazeno zpozdeni
    for(i=0; i<100; i++){
      portNOP();
    }
    // inkrementace citace IdleHook
    IdleHookCitac++;
}
```

Ve volném čase mikrokontroléru, tedy ve funkci *vApplicationIdleHook* je inkrementována hodnota *IdleHookCitac*. Čím déle mikrokontrolér setrvává ve funkci, tím déle se proměnná inkrementuje.

Každou 1 sekundu se spouští vlákno pro výpočet aktuálního vytížení. Hodnota *IdleHookCitac* se vynásobí konstantou *k* (-0.003017046312) a převede na rozsah $0 \div 100\,\%$.

Konstanta *k* je zjištěna experimentálně a jedná se o převrácenou hodnotu proměnné *IdleHookCitac*, která je zjištěna při měření na zcela nevytíženém mikrokontroléru, tedy při jeho 0 % vytížení.

Naměřené výsledky vytížení se pohybovaly okolo $6 \div 12\,\%$ v závislosti na aktuálním zatížení mikrokontroléru.

10.2 Modul správy energie

Program v modulu správy energie je rozdělen na jednotlivá vlákna dle jejich funkce a jejich periodě volání. Vlákna můžou nabývat hodnoty priority od nejmenší (0) po největší (4). V následujícím přehledu jsou vlákna seřazena dle jejich priority od největší po nejmenší:

- **vCpuTimeThread –** vlákno pro sledování a výpočet vytížení procesoru, voláno s periodou 1 sekunda.
- **vEepromStoreTask –** vlákno pro ukládání důležitých údajů do nonvolatilní paměti s periodou volání 5 sekund.
- **vUartRxTask –** voláno vždy při dokončení příjmu nové zprávy na sériové lince.
- **vUartTxTask –** vlákno pro odesílání zpráv na sériovou linku a voláno s periodou 100 ms.
- **vBatteryManagementTask –** slouží pro sledování stavu a nabíjení (včetně balancování) akumulátoru a volá se periodicky každých 10 ms.
- **vSpeakerTask –** vlákno pro generování zvuků pro reproduktor informujícího o stavu systému, voláno s periodou 20 ms.
- **vLedTask –** vlákno informující o stavu operačního systému FreeRTOS, voláno cyklicky s periodou 500 ms.

Kromě těchto vláken zde běží vlákna pro CANopen komunikaci, tyto jsou ale uživateli skrytá a proto nejsou zahrnuta ani v přehledu. Jejich popis nalezneme v kapitole 9. Komunikace.

Modul správy energie je **CANopen heartbeat consumer** (master), tudíž přijímá zprávy od ostatních CANopen heartbeat producers (slaves), které je vysílají s periodou 500 ms. Pokud takováto zpráva od producera nedorazí do časového limitu jedné sekundy, je daný CANopen modul považován za nečinný, chybný.

Slovník objektů využívaných modulem správy energie je pro svou velikost uveden v přílohách. Data týkající se správy energie jsou následně ze slovníku objektů uložena ve struktuře *BatteryModule* a data týkající se komunikace podřízených systému s nadřazeným systémem i.Mx31 jsou uložena ve struktuře *iMX31Data*. Struktura *BatteryModule* obsahuje:

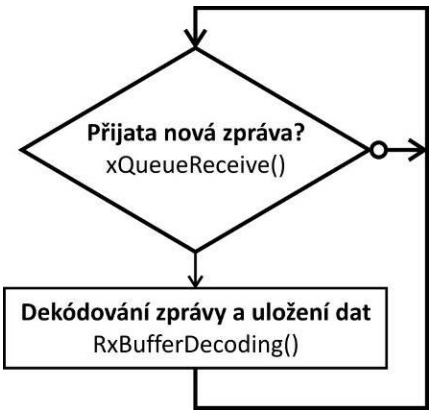
- data pro sledování stavu a nabíjení akumulátoru, dodaná a odebraná kapacita akumulátoru, doba běhu akumulátoru, atd.
- napětí jednotlivých článků akumulátoru, data potřebná pro balancování

Protože se ke struktuře *iMX31Data* a *BatteryModule* přistupuje z více vláken, je potřeba k nim řešit výlučný přístup pomocí nástroje mutex. Jakýkoliv zápis nebo čtení dat struktur je chráněn mutexem *xMutexGlobalData*.

V následujících podkapitolách budou podrobněji popsána jednotlivá vlákna modulu správy energie.

10.2.1 Vlákno vUartRxTask

Vlákno sloužící pro příjem a dekódování obsahu zpráv přijímaných z nadřazeného systému i.MX31. Jednotlivé Byty zprávy jsou přijímány v přerušení od příjmu na sériové lince a celá zpráva pak uložena do fronty zpráv. Operační systém zjistí, že fronta zpráv obsahuje alespoň jednu zprávu a spustí se vykonávání vlákna *vUartRxTask*. Ta obsahuje funkci *xQueueReceive(),* na které se čeká tak dlouho, dokud není ve frontě alespoň jedna zpráva. Jakmile je zpráva přijata, postoupí se k jejímu zpracování, dekódování a uložení dat na jejich místa v paměti. Algoritmus lze vidět na obrázku Obr. 44.



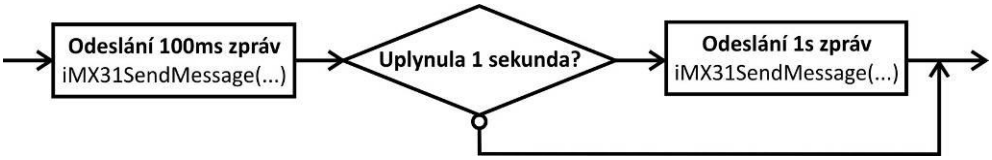
Obr. 44 Vlákno vUartRxTask

Vlákno je voláno tak často, jak často jsou přijímány zprávy z nadřazeného systému. Jelikož je čekání na funkci *xQueueReceive()* řešeno samotným operačním systémem, není na rozdíl od metody cyklického dotazování procesor zbytečně zatěžován. Více k formátu zpráv a jejich příjmu v kapitole *9.1.2 Vyšší vrstva pro rozhraní UART*.

Protože se ve funkci *RxBufferDecoding()* zapisuje do globálních dat, které jsou přístupné i z jiných vláken, je přístup ochráněn semaforem *xMutexGlobalData*.

10.2.2 Vlákno vUartTxTask

Vlákno provádí cyklické odesílání zpráv s periodou 100 ms do nadřazeného systému pomocí funkce iMX31SendMessage. Protože se některá data nemusí odesílat tak často, je ve vláknu ještě čítač 100ms intervalů a pokud je čítač roven hodnotě 10, uplynula tedy 1 sekunda, odešle se další blok méně časově kritických dat.

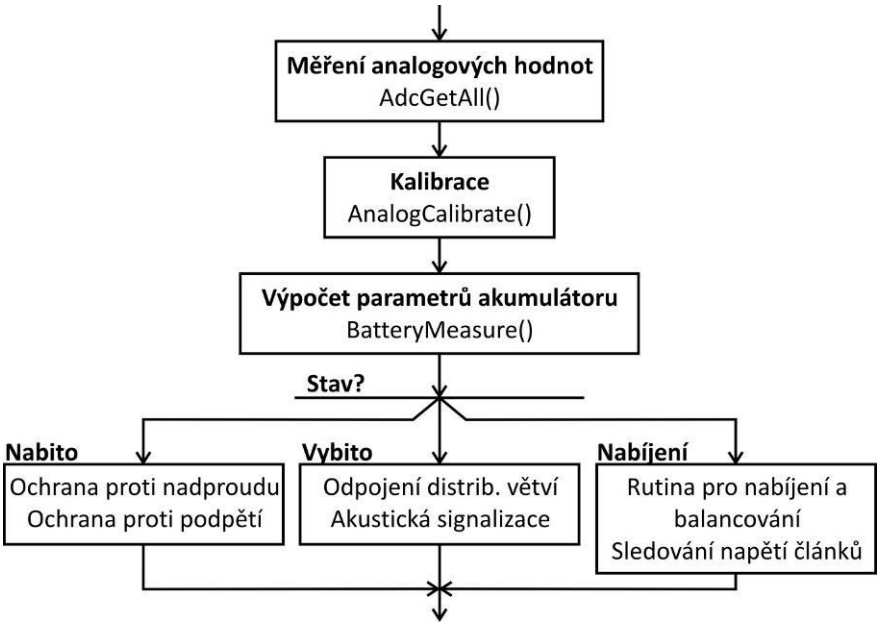


Obr. 45 Vlákno vUartTxTask

Vstupními parametry funkce *iMX31SendMessage(...)* jsou identifikátor zprávy a ukazatel na data k odeslání.

10.2.3 Vlákno vBatteryManagementTask

Vlákno provádí cyklické sledování stavu akumulátoru. Je voláno s periodou 10 ms. Je to nejdůležitější vlákno, ve kterém jsou umístěny funkce týkající se správy akumulátoru. Algoritmus programu vlákna lze vidět na obrázku Obr. 46.



Obr. 46 Vlákno vBatteryManagementTask

- **AdcGetAll() –** Jako první se ve vláknu provede odměření všech analogových hodnot. Mezi ně patří:
  - Nabíjecí napětí a nabíjecí proud
  - Napětí akumulátoru a odebíraný proud
  - Napětí na jednotlivých článcích 6článkového akumulátoru (pro potřeby balancování)

- **AnalogCalibrate() –** Naměřené analogové hodnoty je potřeba kalibrovat a převést je na jejich reálné fyzikální rozsahy. 10bitová hodnota pokrývá u použitých proudových čidel skutečný rozsah $-20 \div 20\ A$ a u měření napětí rozsah $0 \div 30\ V$. Je tedy potřeba nalézt koeficienty K, Q a ty pak použít v kalibrační funkci. Kromě toho, vlivem různých nepřesností, bylo potřeba zkontrolovat skutečný proud odečtený z ampérmetru, resp. napětí z voltmetru a porovnat je s hodnotami naměřenými a případné korekce zanést rovněž do vztahu (22). Použitá kalibrace se nazývá dvoubodová kalibrace.

- **BatteryMeasure –** V této funkci se počítá aktuální spotřeba vozidla, doba běhu akumulátoru a odebraná kapacita při vybíjení. Tyto hodnoty jsou důležité pro sledování aktuální spotřeby vozidla a výpočet zbývající doby jízdy. Vztahy pro výpočet uvedených hodnot jsou uvedeny v kapitole *6.2.2 Sledování stavu akumulátoru*.

Následně se program větví podle toho, v jakém režimu provozu je akumulátor. Jedná se o režimy Nabito, vybito a nabíjení.

Pokud je akumulátor vybit, odpojí se všechny distribuční napájecí větve pro ostatní moduly a je aktivována zvuková signalizace upozorňující na nízkou úroveň napětí akumulátoru. Za nízkou úroveň je zde brána hodnota napětí akumulátoru menší než 20 V. Pokud je akumulátor vybit, měl by operátor co nejrychleji (do 15 minut) připojit zdroj nabíjení nebo vozidlo vypnout. Z akumulátoru je totiž stále odebírána energie pro modul správy energie, který nedokáže sám sebe vypnout.

Po připojení zdroje pro nabíjení se automaticky přejde do stavu, ve kterém se řídí nabíjení, balancují články akumulátoru a sledují se jejich jednotlivá napětí pro detekci ukončení nabíjení. Rovněž se zde vypočítává hodnota dodané kapacity v mAh a doba nabíjení v sekundách.

Po ukončení nabíjení nebo odpojení zdroje nabíjení se automaticky přechází do stavu *Nabito*, ve kterém se monitoruje aktuální napětí a proud akumulátoru pro potřeby ochrany proti podpětí a nadproudu.

10.2.4 Vlákno vEepromStoreTask

Vlákno provádí cyklické ukládání důležitých dat paměti EEPROM s periodou 5 sekund a tím je zálohuje pro případ odpojení napájení. Mezi ukládaná data patří hlavně kapacita akumulátoru dodaná při nabíjení, velikost odebrané kapacity při běhu z akumulátoru, doba běhu akumulátoru a doba posledního nabíjení akumulátoru. Jedná se tedy o data týkající se pouze správy energie, konkrétně zjištění zbývající doby jízdy, procenta nabití akumulátoru atd.

10.2.5 Vlákno vSpeakerTask

Vlákno prováděné cyklicky s periodou 20 ms a slouží pro zvukovou signalizaci stavu modulu správy energie a potažmo celého auta. Jsou implementovány signály pro následující události (řazeno dle priority od největší po nejmenší):

- **BatteryDischarged** – tón upozorňující na vybití akumulátoru; vozidlo je potřeba neprodleně nechat nabíjet nebo vypnout napájení
- **CommunicationOffline** – upozornění na výpadek komunikace modulu správy energie s nadřazeným systémem i.MX31; podřízený systém přechází do nouzového režimu, kdy není vozidlu povoleno pokračování v jízdě
- **WifiOffline** – upozornění na výpadek komunikace mezi i.MX31 a vzdáleným řídicím systémem; vozidlu není opět dovoleno pokračovat v jízdě
- **DestinationReached** – tón upozorňující na dosažení požadované cílové destinace
- **Back** – signál upozorňující na couvání vozidla
- **Klakson** – klakson ovládaný uživatelem ze vzdáleného řídicího systému
- **ObstacleDetected** – tón upozorňující na detekovanou překážku před automobilem; ovládáno z nadřazeného systému i.MX31

11 Shrnutí dosažených výsledků

Hlavní náplní této diplomové práce bylo vytvořit moduly pro správu energie a řízení pohonu.

Pro oba moduly bylo vytvořeno hardwarové řešení, obsahující návrh schémat zapojení a oboustranných desek plošných spojů. Dále byl vytvořen řídicí program v jazyce C.

Modul správy energie slouží pro distribuci elektrické energie ze zdroje k palubní elektronice vozidla. Ta je rozdělena do tří samostatných okruhů. První okruh je tvořen nadřazeným systémem i.MX31, druhý podřízeným systémem modulů připojených na sběrnici CAN. Třetí okruh je pak tvořen pohonem, který odebírá značnou část energie dodávané akumulátorem. Jako zdroj elektrické energie byl zvolen Li-pol akumulátor o jmenovitém napětí 22,2 V a kapacitě 4350 mAh. Výsledná doba jízdy vozidla se pohybovala okolo 2 ÷ 2,5 hodiny při průměrné spotřebě 45 Wh během jízdy. Je zde implementována ochrana proti případnému nadproudu, která se aktivuje při odběrech větších než 10 A. Ta způsobí odpojení všech distribučních větví a opětovné zapnutí vozidla je možné pouze po restartu celého vozidla. Nastavená mez je dostatečná a během testování nedošlo ani jednou k jejímu dosažení. Dále je použita ochrana proti podpětí akumulátoru. Ta je nastavena na 20 V. Pokud napětí akumulátoru klesne pod tuto hranici, je akumulátor považován za vybitý. Pro zjištění míry vybití akumulátoru je použito principu porovnání dodané kapacity při nabíjení a odebrané kapacity při provozu vozidla. Tato metoda poskytuje v praxi uspokojivé výsledky s tolerancí ±2 % z celkové kapacity, což znamená přesnost ±3 minuty do reálného vybití akumulátoru. Pro nabíjení akumulátoru slouží laboratorní zdroj 30 V / 2 A, pro dobíjení například solární článek, jehož napětí musí být rovněž větší než aktuální napětí na akumulátoru. Vybitý akumulátor je nabíjen proudem 2 A po dobu zhruba 2,5 hodiny. Nabíjecí napětí je spolu s nabíjecím proudem měřeno a použito pro potřeby algoritmu nabíjení. Vzhledem k použitému typu akumulátoru Li-pol je zde využit i obvod pro balancování, který byl spolu s algoritmem nabíjení prací jiného studenta v rámci bakalářské práce. Implementace algoritmu do modulu správy energie průzkumného vozidla již byla obsahem této diplomové práce.

Modul správy energie rovněž obstarává funkci datového mostu mezi nadřazeným systémem i.MX31 a podřízenými moduly s mikrokontroléry HCS12. Pro komunikaci s nadřazeným systémem je využito 3V UART rozhraní s komunikační rychlostí 19200 Bd. Pro tuto komunikaci byl vytvořen komunikační protokol, rozdělující data na jednotlivé zprávy, které jsou zasílány v pravidelných 100ms intervalech. Pro komunikaci s podřízeným systémem je využito CAN sběrnice s komunikační rychlostí 125 kbit. Pro tuto velmi spolehlivou sběrnici byl napsán komunikační protokol CANopen. Ten rovněž rozděluje data na jednotlivé zprávy a výměna procesních dat se opět děje ve 100ms intervalech. Výpadek komunikace s kterýmkoliv modulem na sběrnici CAN nebo s jednotkou i.MX31 na komunikačním rozhraní UART je detekován a vozidlo automaticky přechází do nouzového režimu. V něm je zamezeno pojezdu vozidla, aktivována mechanická brzda a stav vozidla je operátorovi akusticky indikován.

Modul pohonu má na starost pohon průzkumného vozidla. Pro pojezd vozidla byl vzhledem k velké účinnosti a dalším výborným vlastnostem vybrán stejnosměrný 150W motor

Maxon se samonosným rotorovým vinutím. Ten spolu s převodovkou s poměrem 4,3:1 tvoří dostatečně dimenzovanou pohonnou jednotku s celkovou rychlostí vozidla 6,5 km/h. Jmenovitý proud motoru je 5 A, jeho průměrná hodnota se při jízdě pohybovala okolo 2,5 A. K motoru je dodáván kvadraturní enkodér s počtem 256 impulsů/ot, jehož výstupem jsou dva obdélníkové signály vzájemně posunuty o 90°. Pro dekódování směru otáčení motoru bylo využito vhodného zapojení klopného obvodu typu D, pro zjištění rychlosti otáčení motoru pak záchytného systému mikrokontroléru. Motor je buzen čtyřkvadrantovým měničem, nazývaným také jako H-bridge. Ten je tvořen čtveřicí výkonových N MOSFET tranzistorů s $R_{DS(on)} = 2\,m\Omega$, které umožňují proud motoru až 35 A při výkonové ztrátě 2,45 W na tranzistoru. H-bridge je buzen předbudičem, který zaručuje potřebnou strmost hran při spínání tranzistorů a také řeší buzení horních N MOSFET tranzistorů. Předbudič je ovládán komplementárním PWM signálem u kterého je hardwarově vyřešen tzv. mrtvý čas, který pohybuje okolo hodnoty 4 µs. PWM signál je akční veličinou vystupující z vytvořeného softwarového PSD regulátoru. Vstupní veličinou jsou pak požadované otáčky z nadřazeného systému a aktuální otáčky z enkodéru. Aby se zamezilo skokové změně požadovaných otáček, je generována náběhová nebo doběhová rampa, která tomuto zamezuje. Regulační smyčka je doplněna ochranou proti proudovému přetížení motoru. Ta nastavena na hranici 5 A a spolehlivě chrání motor proti přetížení.

Modul pohonu komunikuje s okolím prostřednictvím sběrnice CAN s rychlostí 125 kbit, pro kterou byl napsán komunikační protokol CANopen. Ten rozděluje data na jednotlivé zprávy a výměna procesních dat se děje opět ve 100ms intervalech.

Na obou modulech s mikrokontrolérem HCS12 a vnitřní frekvencí sběrnice 25 MHz byl nasazen operační systém reálného času FreeRTOS. Ten umožňuje použití vláken a synchronizačních nástrojů. Použitý operační systém tak umožnil vytvořit periodicky se spouštějící vlákna a vykonávající určitou část řídicího algoritmu, např. nabíjení akumulátoru, regulace otáček motoru, odesílání a příjem zpráv na UARTu apod. Pomocí fronty zpráv zase byly vytvořeny buffery pro přijaté a odesílané zprávy na použitých komunikacích CAN a UART. Dále pomocí mutexu byl vyřešen přístup do sdílené paměti. Systémový tick byl zvolen 1 ms. Po provedení časových měření bylo zjištěno, že nedochází k rapidnímu nárůstu režie při použití operačního systému FreeRTOS. Použití operačního systému navíc usnadnilo mnohdy jinak složité řešení.

12 Závěr

Cílem diplomové práce bylo navrhnout řešení pro správu energie a řízení pohonu mobilního robotického zařízení. Pro model robotického zařízení bylo využito podvozku modelu auta v měřítku 1:7 s pohonem zadních kol. Model byl následně doplněn elektromotorem s převodovkou, zdrojem energie, řídicími moduly a senzory pro snímání okolí. Celková koncepce průzkumného vozidla je pojata jako hierarchický distribuovaný řídicí systém. Na vrcholu hierarchie se nachází řídicí modul s procesorem Freescale i.MX31. Ten komunikuje s podřízenými moduly s použitými mikrokontroléry Freescale HCS12. Na tomto modelu průzkumného vozidla spolupracovali čtyři diplomanti a v rámci svých diplomových prací se svou částí podíleli na celkovém řešení. Hlavní náplní této diplomové práce bylo vytvořit moduly pro správu energie a řízení pohonu.

Pro oba moduly bylo vytvořeno hardwarové řešení, obsahující návrh schémat zapojení a oboustranných desek plošných spojů. Dále byl vytvořen řídicí program v jazyce C.

Cíle diplomové práce se podařilo splnit a vozidlo je nyní v plně funkčním stavu schopné autonomního pohybu. Díky navrženým modulům správy energie a pohonu je umožněn pohyb vozidla a vyřešena otázka napájení palubní elektroniky.

Průběh práce byl intenzivně konzultován s vedoucím diplomové práce a členy týmu.

Do budoucna by se dalo zvážit použití Fuzzy regulátoru pro regulaci otáček pohonu, který by upravoval styl jízdy podle členitosti terénu, údajů z okolí a stavu vozidla. Dále by se mohly na vozidlo namontovat solární panely a ty pak lépe zaintegrovat do správy napájení.

13 Použitá literatura

[1] RICHARD, Barry. *Using the FreeRTOS real time kernel : Practical guide* [online]. [s.l.] : [s.n.], 2009 [cit. 2010-04-24]. Dostupné z WWW: <www.freertos.org>.

[2] RICHARD, Barry. *FreeRTOS reference manual : API functions and configuration options* [online]. [s.l.] : [s.n.], 2009 [cit. 2010-04-24]. Dostupné z WWW: <www.freertos.org>.

[3] CAN in Automation e. V., *CANopen application layer and communication profile : CiA draft standard 301* [online]. 4th edition. [s.l.] : [s.n.], 2002 [cit. 2010-04-24]. Dostupné z WWW: <can-cia.org>.

[4] VACEK, František. *CANopen – vyšší komunikační protokol pro vestavné sítě* [online]. 2004 [cit. 2010-04-25]. Automa. Dostupné z WWW: <http://www.odbornecasopisy.cz/index.php?id_document=32279>.

[5] Motorola, Inc. *MC9S12DP512 : Device user guide* [online]. 2nd edition. [s.l.] : [s.n.], 2005 [cit. 2010-04-24]. Dostupné z WWW: <www.freescale.com>.

[6] MACHÁČEK, Zdeněk; SROVNAL, Vilém. *Popis, identifikace systému a návrh regulátoru pomocí Matlabu v aplikaci Fotbal robotů* [online]. [s.l.] : [s.n.], [200?] [cit. 2010-04-24]. Dostupné z WWW: <http://dsp.vscht.cz/konference_matlab/MATLAB07/prispevky/machacek_srovnal/machacek_srovnal.pdf>.

[7] HLAVA, Jaroslav. *Číslicové PID regulátory* [online]. [s.l.] : [s.n.], [200?] [cit. 2010-04-24]. Dostupné z WWW: <www.fm.tul.cz/~krtsub/fm/par/digitalPID.pdf>.

[8] Atmel Corporation. AVR221 : *Discrete PID controller* [online]. [s.l.] : [s.n.], [200?] [cit. 2010-04-24]. Dostupné z WWW: <http://www.atmel.com/dyn/resources/prod_documents/doc2558.pdf>.

[9] MGM compro. *Vnitřní odpor Lipol článků a jeho vliv na vypínací napětí regulátorů* [online]. [s.l.] : [s.n.], [200?] [cit. 2010-04-24]. Dostupné z WWW: <http://mgm-compro.cz/pdf/vnitrni-odpor-lipol-clanku-a-jeho-vliv-na-vypinaci-napeti-regulatoru.pdf>.

[10] KOCMAN, Stanislav. *Stejnosměrné stroje* [online]. [s.l.] : [s.n.], 2002 [cit. 2010-04-24]. Dostupné z WWW: <http://www.isse.pr-net.cz/materialy/ESP/stejnosmerne.pdf>.

[11] Uzimex. *Motory DC - maxon motor ag - Elektrické pohony - UZIMEX* [online]. [200?] [cit. 2010-04-24]. Katalog motorů DC. Dostupné z WWW: <http://www.uzimex.cz/Sortiment/Elektricke-pohony/maxon-motor-ag/Motory-DC.html>.

[12] RCM Pelikán. *RCM Pelikán* [online]. [200?] [cit. 2010-04-25]. Li-pol akumulátorová baterie. Dostupné z WWW: <http://www.rcm-pelikan.cz/index.php?sec=product&id=29753>.

[13] Hobby Products International Europe Ltd. *HPI Baja 5b SS RC Car* [online]. [200?] [cit. 2010-04-25]. RC Buggy Model. Dostupné z WWW: <http://www.hpieurope.com/kit-info.php?lang=en&partNo=10611>.

[14] Nickel-cadmium battery In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2010, 2003 [cit. 2010-04-25]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Nickel-cadmium_battery>.

[15] Nickel metal hydride battery In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2007, 2007 [cit. 2010-04-25]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Nickel_metal_hydride_battery>.

[16] Lead-acid battery In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2003, 2010 [cit. 2010-04-25]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Lead%E2%80%93acid_battery>.

[17] Lithium-ion battery In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26 March 2003, 23 April 2010 [cit. 2010-04-25]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Lithium-ion_battery>.

[18] Lithium-ion polymer battery In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 26 March 2003, 12 April 2010 [cit. 2010-04-25]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Lithium-ion_polymer_battery>.

[19] Lithium iron phosphate battery In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, December 2006, 14 April 2010 [cit. 2010-04-25]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Lithium_iron_phosphate_battery>.

[20] *Číslicové filtry* [online]. [s.l.] : [s.n.], [200?] [cit. 2010-04-25]. Dostupné z WWW: <http://www.rss.tul.cz/download/cms/11_cisfiltry.pdf>.

[21] *Odhad plovoucího průměru* [online]. 2005-11-10 [cit. 2010-04-25]. Robotika. Dostupné z WWW: <http://robotika.cz/guide/filtering/en>.

[22] *Motory a jejich řízení s MCU : 1.část - typy motorů* [online]. [s.l.] : [s.n.], 2008-02-17 [cit. 2010-04-25]. Dostupné z WWW: <http://automatizace.hw.cz/motory-jejich-rizeni-s-mcu-1-cast-typy-motoru>.

[23] Maxon motor. *Malé stejnosměrné motory* [online]. [s.l.] : [s.n.], 2002-07-18 [cit. 2010-04-25]. Dostupné z WWW: <http://www.uzimex.cz/soubory/20070103_maxon_serial.pdf>.

[24] Maxon motor. *Katalogový list : Maxon DC motor RE40* [online]. [s.l.] : [s.n.], [2005] [cit. 2010-04-25]. Dostupné z WWW: <http://test.maxonmotor.com/docsx/Download/catalog_2005/Pdf/05_083_e.pdf>.

[25] Maxon motor. *Katalogový list : Planetová převodovka GP 42 C* [online]. [s.l.] : [s.n.], [2005] [cit. 2010-04-25]. Dostupné z WWW: <http://test.maxonmotor.com/docsx/Download/catalog_2005/Pdf/05_224_e.pdf>.

[26] Maxon motor. *Katalogový list : Enkodér MR typ L 256* [online]. [s.l.] : [s.n.], [2005] [cit. 2010-04-25]. Dostupné z WWW: <http://www.treffer.com.br/produtos/maxon/tacho/pdf/239.pdf>.

[27] *BRT Phileas* [online]. [200?] [cit. 2010-04-25]. Phileas; Bus Rapid Transit in Eindhoven. Dostupné z WWW: <http://connectedcities.eu/showcases/phileas.html>.

[28] *DARPA project* [online]. [200?] [cit. 2010-04-25]. DARPA Urban Challenge. Dostupné z WWW: <http://www.darpa.mil/grandchallenge/index.asp>.

[29] *Towards advanced road transport for the urban environment* [online]. 2010 [cit. 2010-04-25]. CityMobil. Dostupné z WWW: <http://www.citymobil-project.eu/>.

14 Seznam příloh

Příloha I	-	Slovník komunikačních objektů CANopen
Příloha II	-	Tabulka zpráv pro komunikaci s nadřazeným systémem i.MX31
Příloha III	-	Schéma zapojení modulu pohonu
Příloha IV	-	DPS – modul pohonu
Příloha V	-	Schéma zapojení modulu správy energie
Příloha VI	-	DPS – modul správy energie
Příloha VII	-	Fotodokumentace

Přiložené DVD obsahuje:

1. Diplomová práce.pdf – vlastní práce
2. Diplomová práce.doc – vlastní práce
3. /Fotodokumentace – Fotodokumentace průzkumného vozidla (389 MB)
4. /Videa – Video ukázky průzkumného vozidla v HD kvalitě (720p) (567 MB)
5. /Obrázky – Obrázky použité v textu diplomové práce (13,6 MB)
6. /Obrázky/Soubory_COREL – Obrázky s příponou *.cdr spustitelné v Corelu (2,45 MB)
7. /Použitá literatura – Záloha použité literatury (28,7 MB)
8. /Použitá literatura/DATASHEETS – Katalogové listy použitých el. součástek (17,7 MB)
9. /Použitá literatura/DATASHEETS/maxon_vypocet_momentu.xls – Návod firmy Uzimex pro výpočet potřebného krouticího momentu motoru
10. /Schémata zapojení a DPS/Modul_pohonu – Schéma zapojení a podklady pro výrobu DPS modulu pohonu (2,28 MB)
11. /Schémata zapojení a DPS/Modul_spravy_energie – Schéma zapojení a podklady pro výrobu DPS modulu správy energie (2,94 MB)
12. /Software/Modul_pohonu – Projekt modulu pohonu spustitelný v programu CodeWarrior
13. /Software/Modul_spravy_energie – Projekt modulu správy energie spustitelný v programu CodeWarrior

Příloha I - Slovník komunikačních objektů CANopen

Modul pohonu									
Zpráva	**Identifikátor (COB-ID)**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
TPDO1 + DM	0x185	AktualniOtackyMotoru [ot/min] 2001H/1		ProudMotoru [mA] 2001H/2		Brzda [0,1] 2001H/3	SmerOtaceni [0,1] 2001H/4		
RPDO1 + DM	0x205	OtackyMotoru [ot/min] 2002H/1		AktivaceBrzdy 2002H/2					
TPDO1 + BM	0x184	NabijeciNapeti [mV] 2003H/1		NabijeciProud [mA] 2003H/2		NapetiBaterie [mV] 2003H/3		OdebiranyProud [mA] 2003H/4	

Legenda	
	Vysíláno z modulu na síť
	Přijímáno ze sítě do modulu
BM	Battery Management – modul správy energie
DM	Drive Module – modul pohonu
MSp	Modul světel - přední
MSz	Modul světel - zadní
MP	Modul polohy

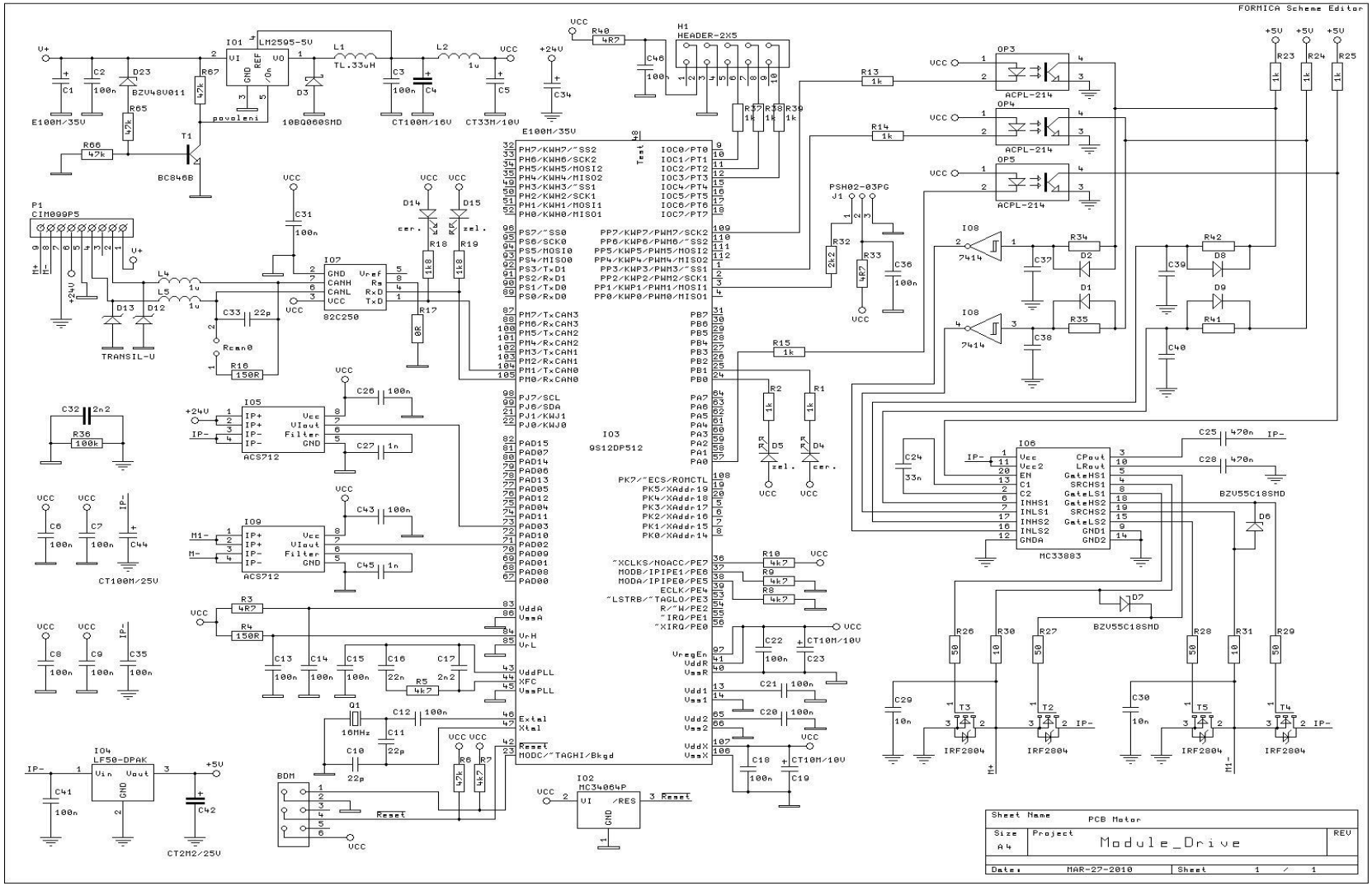
Modul správy energie									
Zpráva	**Identifikátor (COB-ID)**	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**
TPDO1 + BM	0x184	NabijeciNapeti [mV] 2001H/1		NabijeciProud [mA] 2001H/2		NapetiBaterie [mV] 2001H/3		OdebiranyProud [mA] 2001H/4	
RPDO1 + MSp	0x202	Tranzistory [3bit] 2002H/1	Natočení kol [°] 2002H/2	VychoziNatoceniVozidla 2002H/3					
RPDO1 + MSz	0x203	Tranzistory [3bit] 2003H/1	Natočení kol [°] 2003H/2						
RPDO3 + MP	0x401	PocatekMapy (ddmm.mmmmN) 2004H/1				PozatekMapy (dddmm.mmmmE) 2004H/2			
RPDO4 + MP	0x501	Výchozí Poloha X [mm] 2005H/1				Výchozí Poloha Y [mm] 2005H/2			
RPDO1 + DM	0x205	OtackyMotoru [ot/min] 2010H/1		AktivaceBrzdy 2010H/2					
TPDO1 + DM	0x185	AktualniOtackyMotoru [ot/min] 2006H/1		ProudMotoru [mA] 2006H/2		Brzda [0,1] 2006H/3	SmerOtaceni [0,1] 2006H/4		
TPDO1 + MSp	0x182	Ultrazvuk [mm] 2007H/1		IR0 (levý) [mm] 2007H/2		IR1 (pravý) [mm] 2007H/3			
TPDO2 + MSp	0x282	Osvětlení [-] 2008H/1	Nadproud [0,1] 2008H/2	Teplota [°C] 2008H/3					
TPDO4 + MSp	0x482	f=0,addr=nodeID -	index -		subindex -	jeden z 64 různých objektů TPA81 2009H/(1-64)			
TPDO1 + MSz	0x183	Ultrazvuk [mm] 200AH/1		IR2 (pravý) [mm] 200AH/2		IR3 (levý) [mm] 200AH/3			
TPDO2 + MSz	0x283	Osvětlení [-] 200BH/1	Nadproud [0,1] 200BH/2	Teplota [°C] 200BH/3					
TPDO1 + MP	0x181	Poloha X [mm] 200CH/1				Poloha Y [mm] 200CH/2			
TPDO2 + MP	0x281	G - osa X [raw] 200DH/1	G - osa Y [raw] 200DH/2	G - osa Z [raw] 200DH/3	Plyn [%] 200DH/4	Tlak[kPa] 200DH/5	Natočení [°] 200DH/6		
TPDO3 + MP	0x381	hhmmss\0 200EH/1							
TPDO4 + MP	0x481	PolohaAuta (ddmm.mmmmN) 200FH/1				PolohaAuta (dddmm.mmmmE) 200FH/2			

Příloha II - Tabulka zpráv pro komunikaci s nadřazeným systémem i.MX31

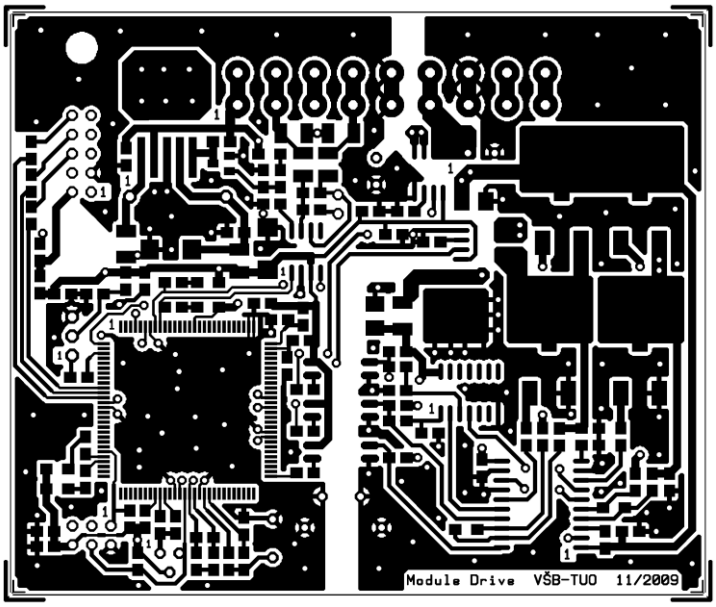
Identifikátor zprávy	Typ	Datový typ	Délka [Byte]	Jednotka	Popis
POLOHAX	0	INT32	5	[mm]	Absolutní poloha na mapě - osa x
POLOHAY	1	INT32	5	[mm]	Absolutní poloha na mapě - osa y
NATOCENIVOZIDLA	2	UINT16	3	[°]	Natočení vozidla vůči magnetickému severu
PREDNIULTRAZVUK	3	INT16	3	[mm]	Přední ultrazvuk
ZADNIULTRAZVUK	4	INT16	3	[mm]	Zadní ultrazvuk
ANALOGOVEDALKOMERY	5	UINT16 pole[4]	9	[mm]	Analogové dálkoměry - 2 přední, 2 zadní
GPSSOURADNICE	6	INT32 pole[2]	9	[°]	GPS souřadnice aktuální polohy vozidla
GPSCAS	7	INT8 pole[7]	8	[-]	Aktuální čas ze signálu GPS
IRKAMERA	8	UINT8 pole[31][8]	249	[°C]	Teplotní profil prostoru před vozidlem
TEPLOTA	9	UINT8	2	[°C]	Teplota okolí
AKCELEROMETRX	10	INT8	2	[-]	Hodnota z akcelerometru - osa x
AKCELEROMETRY	11	INT8	2	[-]	Hodnota z akcelerometru - osa y
AKCELEROMETRZ	12	INT8	2	[-]	Hodnota z akcelerometru - osa z
CIDLOOSVETLENI	13	INT8	2	[-]	Přední čidlo intenzity osvětlení
POCATEKMAPY	14	INT32 pole[2]	9	[°]	GPS souřadnice levého dolního rohu mapy
NATOCENIKOL	16	INT8	2	[°]	Úhel natočení předních kol vozidla
AKTUALNIOTACKYMOTORU	17	UINT16	3	[ot/min]	Aktuální otáčky motoru
PROUDMOTORU	18	UINT16	3	[mA]	Aktuální proud motoru
BRZDA	19	INT8	2	boolean	Stav brzdy
MODBATERIE	20	INT8	2	enum	Nabitá, vybitá, nabíjení
STAVBATERIE	21	INT8	2	[%]	Zbývající kapacita akumulátoru v procentech
AKTUALNISTAVMODULU	22	INT8 pole[3]	4	enum	Připojen, odpojen, chyba

Identifikátor zprávy	Typ	Datový typ	Délka [Byte]	Jednotka	Popis
NABIJECIPROUD	23	UINT16	3	[mA]	Nabíjecí proud
NABIJECINAPETI	24	UINT16	3	[mV]	Nabíjecí napětí
ODEBIRANYPROUD	25	UINT16	3	[mA]	Odebíraný proud
NAPETIBATERIE	26	UINT16	3	[mV]	Napětí akumulátoru
DOBADOVYBITI	27	UINT16	3	[s]	Odhadovaný čas do vybití akumulátoru
OTACKYMOTORU	28	INT16	3	[ot/min]	Požadované otáčky motory
AKTIVACEBRZDY	30	INT8	2	boolean	Požadavek na aktivaci brzdy
STAVMODULU	31	INT8 pole[3]	4	enum	-
VYCHOZIPOZICEX	40	INT32	5	[mm]	Výchozí pozice - osa x
VYCHOZIPOZICEY	41	INT32	5	[mm]	Výchozí pozice - osa y
TOTALSTOP	44	INT8	2	boolean	Nouzové zastavení vozidla
SENDISALIVE	45	INT8	2	null	Požadavek na stav komunikace
RECEIVEISALIVE	46	INT8	2	null	Odpověď na stav komunikace
ZAPNUTISVETEL	50	INT8	2	boolean	Požadavek na zapnutí světel
PLYN	51	UINT8	2	[-]	Údaj z čidla plynu
TLAK	52	UINT16	3	[-]	Údaj z čidla tlaku
CIDLOOSVETLENI2	53	INT8	2	[-]	Zadní čidlo intenzity osvětlení
ZKRAT1	54	UINT8	2	boolean	Zkrat na výstupech předního modulu
ZKRAT2	55	UINT8	2	boolean	Zkrat na výstupech zadního modulu
VYCHOZINATOCENIVOZIDLA	57	INT16	3	[°]	Výchozí natočení vozidla vůči mag. severu
KLAKSON	58	UINT8	2	enum	Zvuková signalizace

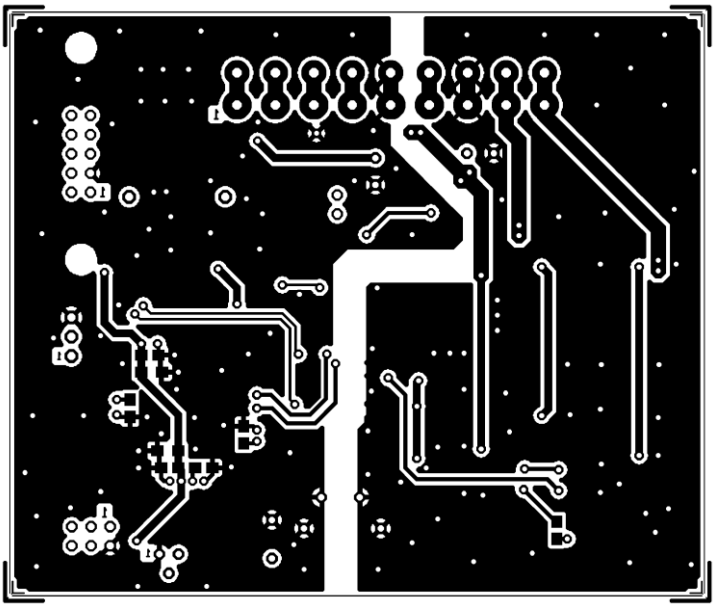
Příloha III - Schéma zapojení modulu pohonu



Příloha IV - DPS – modul pohonu

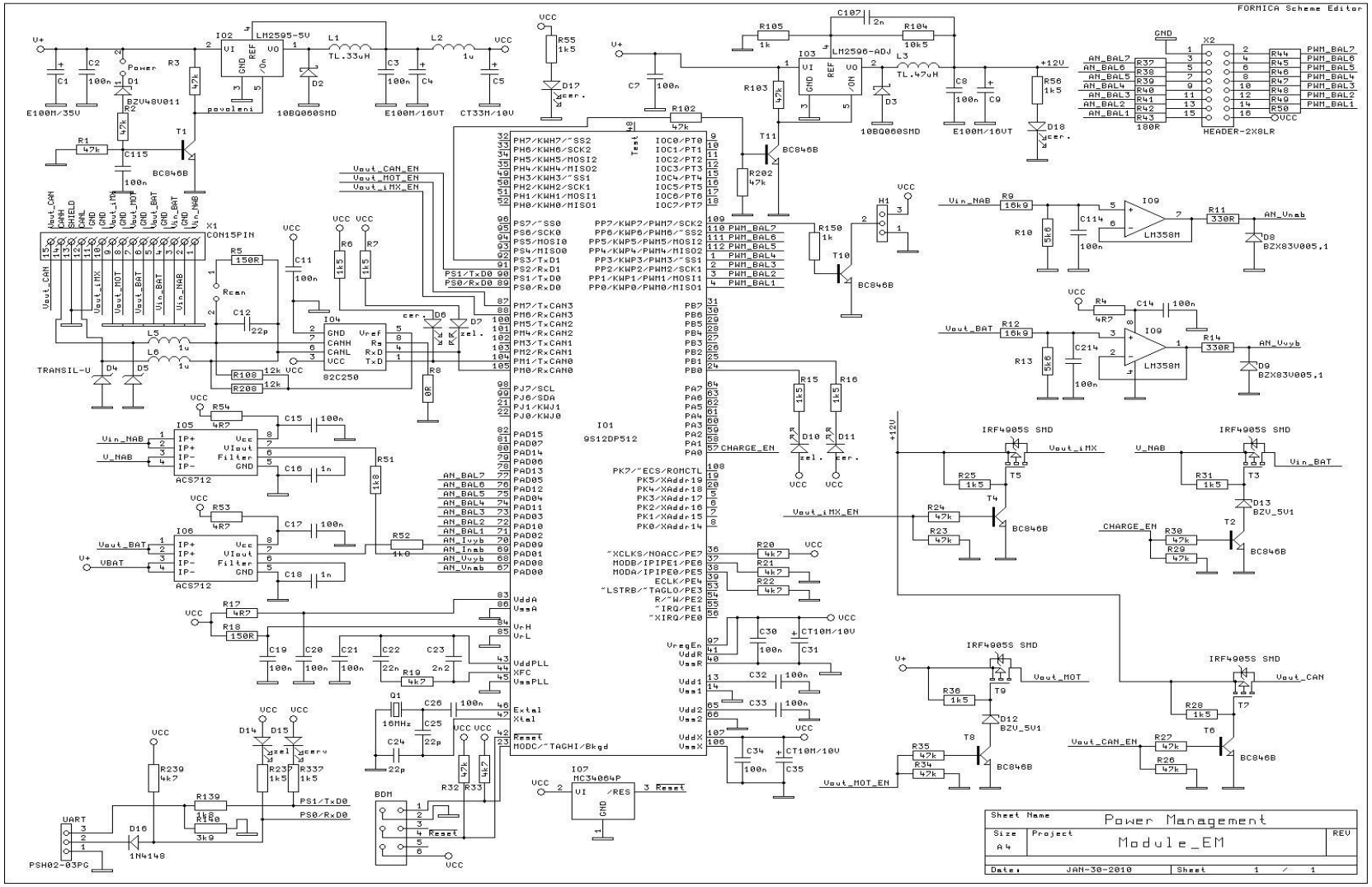


DPS modulu pohonu – strana součástek (TOP), měřítko 1:1

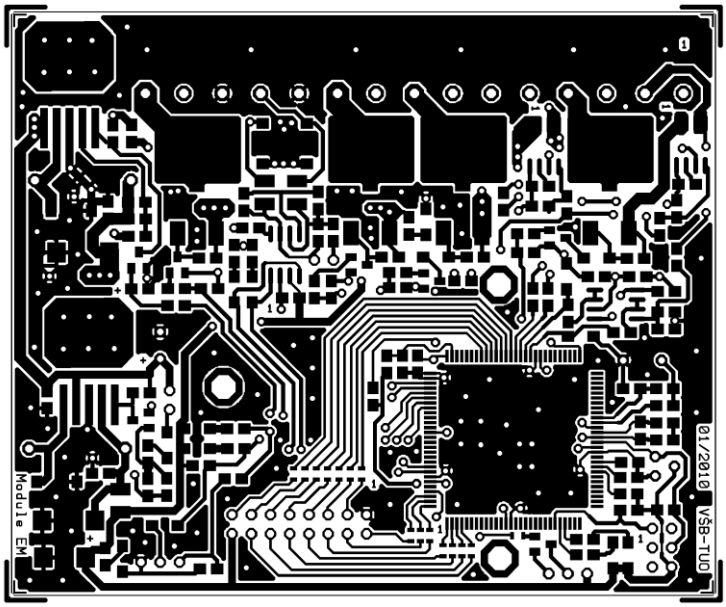


DPS modulu pohonu – strana spojů (BOTTOM), měřítko 1:1

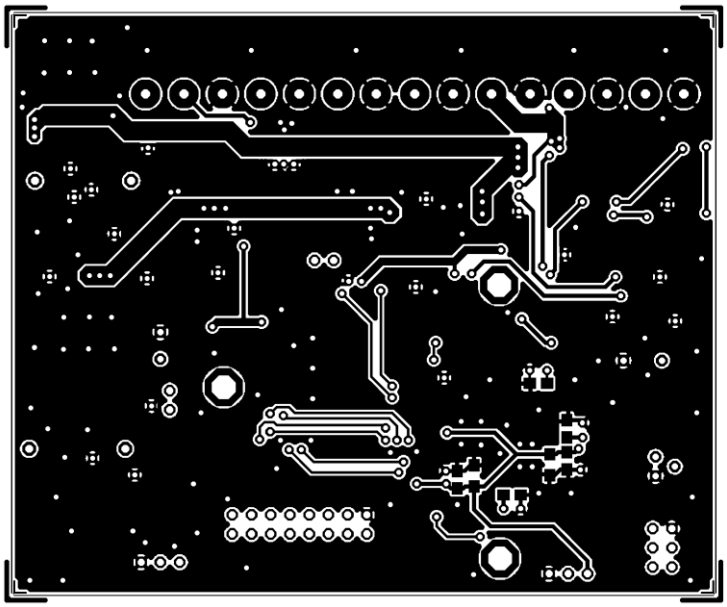
Příloha V - Schéma zapojení modulu správy energie



Příloha VI - DPS – modul správy energie

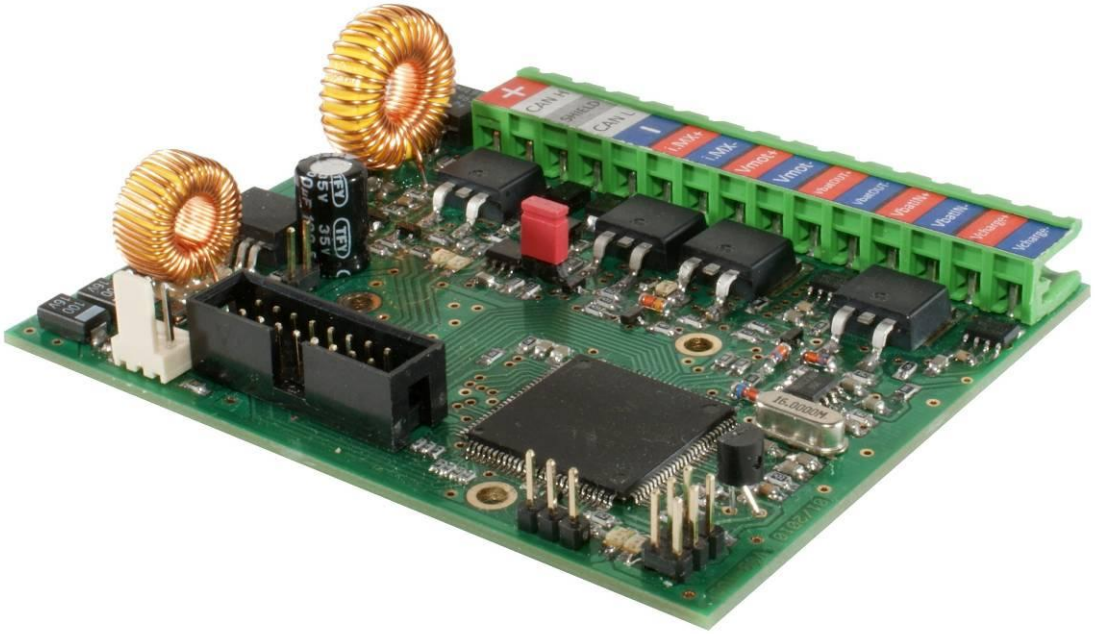


DPS modulu správy energie – strana součástek (TOP), měřítko 1:1

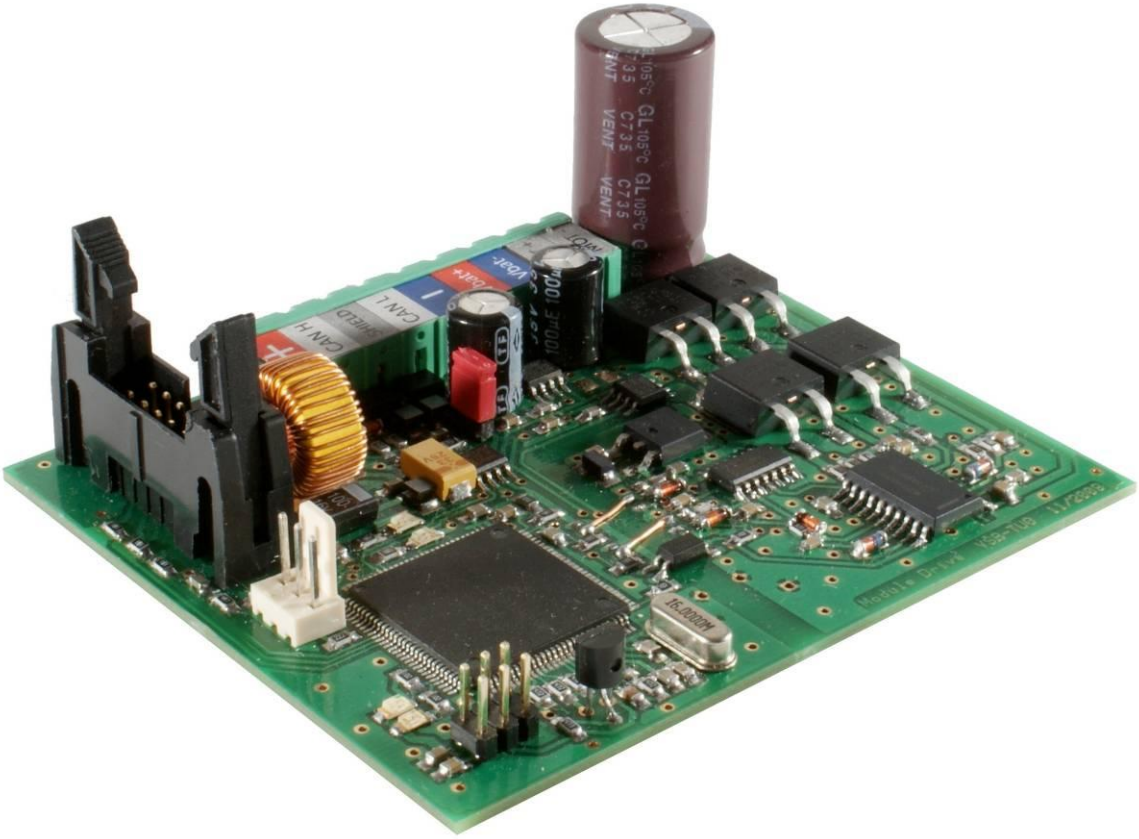


DPS modulu správy energie – strana spojů (BOTTOM), měřítko 1:1

Příloha VII - Fotodokumentace



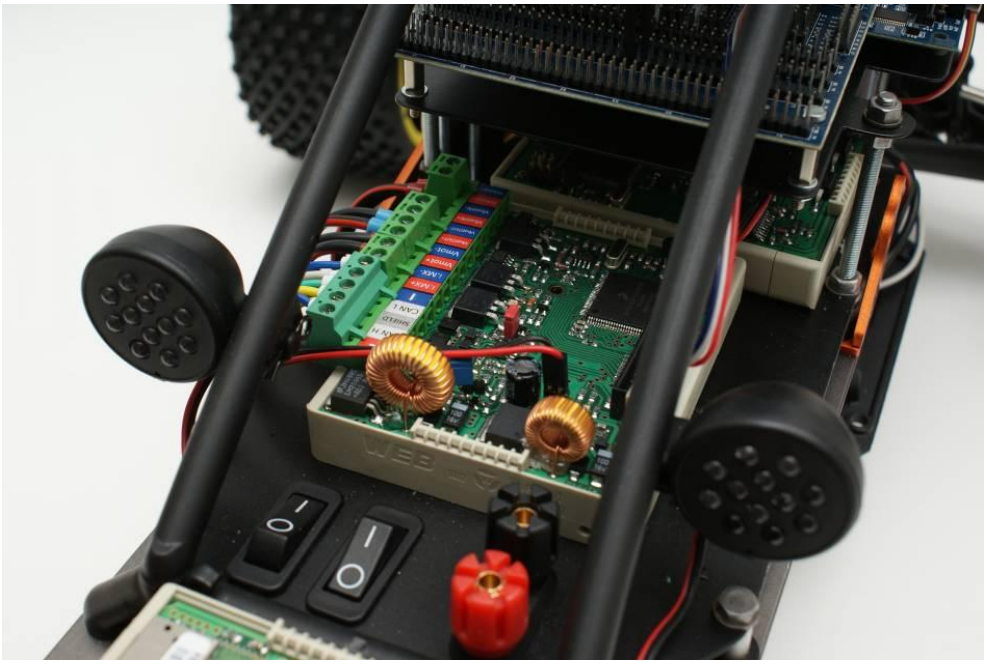
Deska plošných spojů – Modul správy energie



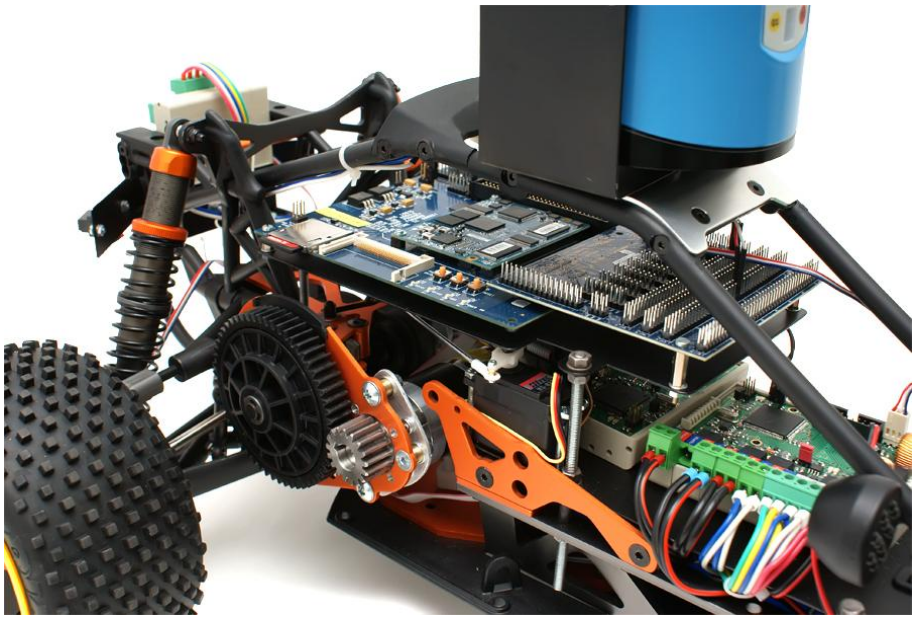
Deska plošných spojů – Modul pohonu

Motor Maxon s převodovkou a enkodérem, použitý pro pohon vozidla

Li-pol akumulátor použitý ve vozidle



Detailní pohled na umístění modulu správy energie



Detailní pohled na pohon vozidla, modulu pohonu a správy energie

Celkový pohled na průzkumné vozidlo po osazení všemi moduly a senzorikou